WEIXING DIANZIJISUANJI JIAOCHENG

微型电子计算机教程

张凯夫 编



R1微型计算机读本

广东科技出版社

微型电子计算机教程

R1微型计算机读本

张凯夫　编

Retro Workshop
果粉工作室

广 东 科 技 出 版 社

内 容 简 介

这是一本学习使用微型计算机的入门书。内容分为三部分：第一部分介绍BASIC算法语言、BASIC语言程序的初步知识及R1微型计算机的上机操作方法；第二部分介绍BASIC程序的各种语句及其编写程序的基本方法；第三部分介绍上机实验及操作指导。各章还附有习题和答案。

本书可作为中学、中专学校或计算机普及培训班初学BASIC语言的教材，也适合具有中等文化程度的读者作自学读本。

微型电子计算机教程

R1微型计算机读本

张 凯 夫 编

*

广 东 科 技 出 版 社 出 版

广 东 省 新 华 书 店 发 行

广 东 新 华 印 刷 厂 印 刷

787×1092毫米 32开本 9.75印张 200,000字

1985年5月第1版 1985年5月第1次印刷

印数1—60,000册

统一书号 15182·115 定价 1.50元

出 版 说 明

《微型电子计算机教程》（以下简称教程）是根据目前中专、中技和职大的教学要求编写的。在编写过程中注意按教学和培训的需要来安排内容，力求概念简明易懂，突出使用和操作技巧，便于学习者自行复习，同时注意与实践相结合。《教程》中收入的许多程序实例都能在实践中直接运用，在学习中起举一反三的作用。

本《教程》首批安排出版的选题共有六种，以目前使用范围较广的TRS—80、APPLE Ⅱ（苹果2型）、R1等机型为教学机，分别介绍各型微型机的BASIC语言的编程知识和上机操作方法、系统结构原理和用汇编语言编程方法等内容。这六种选题是：（1）《微型计算机基础》；（2）《TRS—80微型计算机BASIC Ⅱ语言与磁盘操作系统》；（3）《APPLE Ⅱ微型计算机BASIC语言与磁盘操作系统》；（4）《APPLE Ⅱ微型计算机系统结构》；（5）《APPLE Ⅱ微型计算机实习操作指导》；（6）《R1微型计算机读本》。以后根据教学的需要将补充若干选题。其中，《微型计算机基础》是公共基础课教材，其余各书可由各教学单位根据所采用的教学机型号，结合课程设置和教学要求选用。

参加这套《教程》编写工作的作者是大学的教授、讲师及中专老师，他们曾为几十期培训班讲课，积累了微机教学的经验。《教程》中的六个选题就是在他们讲课的基础上经过认真修改补充后写成的。

本《教程》可供中等专业学校和中级技工学校、职工大学和业余大学开设微型计算机课时作参考教材使用，亦可供各企事业单位和大专院校开办微型机短期培训班时用作教材，还可供广大读者用作自学读本。

前　言

R1 微型计算机具有结构简单，价格便宜，操作方便等特点，是我国引进生产并具有较大产量的一种普及式微型计算机。本书选用它作为教学机，正是为了适应初学使用微型机的广大读者的需要，使之容易入门，易于实践。

本书内容分三部分，共十三章。第一部分是初步介绍 BASIC 语言程序及 R1 机的上机操作方法，使初学者对 BASIC 语言程序的全过程首先有一个完整的概念，并学会上机操作的方法，为学习后面课程打下良好的基础。第二部分介绍 BASIC 程序的各种语句及编写程序的基本方法，使读者能够循序渐进地掌握各种语句的格式、功能，并学会编写一些简单的程序。第三部分的内容是实验、操作指导。实验内容重点突出了语句的特点及其使用方法。

本书是在多次教学实践的基础上整理编写而成的。笔者力求文字通俗、精练，把问题阐述清楚。书中的例题、习题解答和各种程序，都经过上机操作验证。

各种不同类型的微型计算机所用的 BASIC 语言虽然有所差别，但只是大同小异。因此，学习了本书之后，再参阅所用计算机的使用说明书，就不难掌握其它各种微机的使用方法了。

参加本书编写的有梁学明同志。

承蒙深圳大学杨金标副教授在百忙中审校了本书，罗初

东讲师对本书提出了许多宝贵的意见，在此谨致以衷心的感谢。

由于编写时间匆促，书中难免存在缺点和错误，敬请读者批评指正。

编　者

Retro Workshop 果粉工作室

目　录

第一章　概述 …………………………………………… (1)

第一节　R 1 微型计算机系统的基本结构 …………………… (1)

第二节　R 1 微型计算机的技术性能 ………………………… (4)

第三节　机器语言和算法语言 ………………………………… (7)

第二章　BASIC语言程序的构成及上机操作方法 ……………… (10)

第一节　BASIC 语言的特点及构成规则 ……………………… (10)

第二节　R 1 微型计算机的键盘 ……………………………… (15)

第三节　上机操作方法 ………………………………………… (19)

第四节　键盘命令 ……………………………………………… (24)

第五节　源程序的删改 ………………………………………… (28)

第六节　BASIC 语言中数的表示方法 ………………………… (32)

第七节　BASIC 语言中的标准函数 …………………………… (34)

第八节　变量、运算符、运算规则和表达式 …………………… (36)

第九节　键盘运算 ……………………………………………… (39)

习　题 ………………………………………………………… (43)

第三章　赋值语句和输入输出语句 ……………………………… (44)

第一节　打印语句 ……………………………………………… (44)

第二节　赋值语句 ……………………………………………… (51)

第三节　键盘输入语句 ………………………………………… (56)

第四节　读数语句、置数语句和恢复数据区语句 ……………… (61)

习　题 ………………………………………………………… (67)

第四章　无条件转向语句和条件转向语句 ……………………… (70)

第一节　无条件转向语句 ……………………………………… (70)

第二节　条件转向语句 ……………………………………（72）
习　题 ……………………………………………………（81）
第五章　如何编写程序 ……………………………………（83）
第一节　编程的步骤 ……………………………………（83）
第二节　框图（程序流程图） …………………………（84）
第三节　程序的优化考虑 ………………………………（91）
第四节　注释语句 ………………………………………（97）
第六章　程序控制的有关语句 …………………………（99）
第一节　循环语句 ………………………………………（99）
第二节　转子语句和返回语句 …………………………（113）
第三节　暂停语句 ………………………………………（117）
习　题 ……………………………………………………（118）
第七章　数组和下标变量 ………………………………（122）
第一节　单下标变量 ……………………………………（122）
第二节　双下标变量 ……………………………………（132）
习　题 ……………………………………………………（138）
第八章　字符串 …………………………………………（141）
第一节　字符串变量 ……………………………………（141）
第二节　字符串的输入和输出 …………………………（143）
第三节　字符串运算 ……………………………………（146）
第四节　子字符串 ………………………………………（151）
第五节　LEN〈字符串〉函数……………………………（154）
第六节　VAL〈字符串〉函数和STR＄〈算术表达式〉
函数 ………………………………………………（156）
第七节　字符串下标变量 ………………………………（158）
习　题 ……………………………………………………（161）
第九章　函数 ……………………………………………（164）
第一节　取整函数及其应用举例 ………………………（164）
第二节　随机函数及其应用举例 ………………………（175）

第三节　打印格式函数 ……………………………………… (178)
第四节　AT 函数 ……………………………………………… (184)
习　题 ………………………………………………………… (192)
第十章　PLOT语句、UNPLOT语句、PAUSE语句和SCROLL语句 ……………………………………………… (195)
第一节　PLOT（画图）语句 ………………………………… (195)
第二节　UNPLOT（消象元）语句 …………………………… (199)
第三节　PAUSE（暂停）语句 ……………………………… (200)
第四节　SCROLL（滚动）语句 ……………………………… (201)
第十一章　其它语句和函数 ……………………………… (204)
第一节　INKEY＄函数和游戏棒 …………………………… (204)
第二节　R1机的字符集、CODE函数和CHR＄函数 ……… (207)
第三节　POKE语句和PEEK函数 ………………………… (209)
第四节　SLOW和FAST状态 ………………………………… (211)
第十二章　R1微型机和音乐 ……………………………… (214)
第一节　BEEP命令和NO BEEP命令 ……………………… (214)
第二节　MUSIC（音乐）语句 ……………………………… (215)
第三节　TEPMO语句 ………………………………………… (218)
第四节　SOUND语句 ………………………………………… (219)
第十三章　实验、操作指导 ……………………………… (221)
实验一　特殊功能键和键盘命令的操作 ………………… (221)
实验二　源程序的删改 ……………………………………… (223)
实验三　键盘运算 …………………………………………… (224)
实验四　打印语句 …………………………………………… (226)
实验五　INPUT语句 ………………………………………… (228)
实验六　条件语句、循环语句 ……………………………… (231)
实验七　下标变量 …………………………………………… (233)
实验八　函数、随机函数 …………………………………… (237)
实验九　画图、写汉字 ……………………………………… (240)

实验十　游戏和音乐 ………………………………………………… (246)
实验十一　计算机与磁带机的连接——程序的存贮和程序的装入 ……………………………………………… (249)
附录一　习题参考答案 ……………………………………… (253)
附录二　R1微型机字符表 …………………………………… (272)
附录三　ASCII代码 ………………………………………… (284)
附录四　R1微型机函数和语句一览表 ……………………… (287)
附录五　报告信息表 ………………………………………… (296)
附录六　音符字符表 ………………………………………… (299)

Retro Workshop
果粉工作室

第一章　概　　述

自从1946年在美国制造成功世界上第一台电子数字计算机以来，到现在只不过三十多年时间，电子计算机已经经历了从电子管、晶体管、集成电路和大规模集成电路等几个发展阶段，这种发展速度在科学史上是罕见的。特别是七十年代初期开始发展的微型计算机，几乎每两三年就更新一代产品，到现在为止已发展到第五代产品。第五代中的32位微型机，在性能上已经达到甚至超过了小型机。

由于微型计算机具有体积小、价格低、功能强和使用方便等优点，因而使它具有强大的生命力。现在微型计算机的用途，已不只限于科学计算、自动控制和数据处理等方面，而且开始广泛地进入到文化教育、机关办公、家庭生活和劳动就业等领域，对整个社会正在产生越来越大的影响。估计不用很长时间，微型计算机将普遍地成为我国人民工作、学习和生活中的必需品。

第一节　R1微型计算机系统的基本结构

一、微型计算机的组成

一台微型计算机通常包括五个部分：运算器、控制器、存贮器、输入设备和输出设备。其中，运算器、控制器是采用大规模集成电路技术制成的。它们被集成在一块芯片上，这块芯片称为微处理器（MPU），通常又把它称为中央处

理器（CPU）。内存贮器亦全部采用大规模集成电路的半导体存贮器制成，它分为随机存取存贮器（RAM）和只读存贮器（ROM）。输入、输出设备则通过接口电路(I/O接口)与总线连接。这里，总线的作用是把CPU、RAM、ROM和I/O接口电路连接起来。微处理器、内存贮器和外部设备之间的信息交换，也是通过总线进行的。微型计算机有三种总线：数据总线——专用于传送数据；地址总线——专用于传送地址代码；控制总线——专用于传送控制信号。

将上述微处理器、内存贮器、输入输出接口电路和三种总线组装在一起，就构成了一台微型计算机。再将微型计算机加上外部设备、软件以后，便组成微型计算机系统。

二、R1微型计算机的系统结构

R1微型计算机是香港LAMBDA电子有限公司的首批产品，其系统结构如图1-1所示。

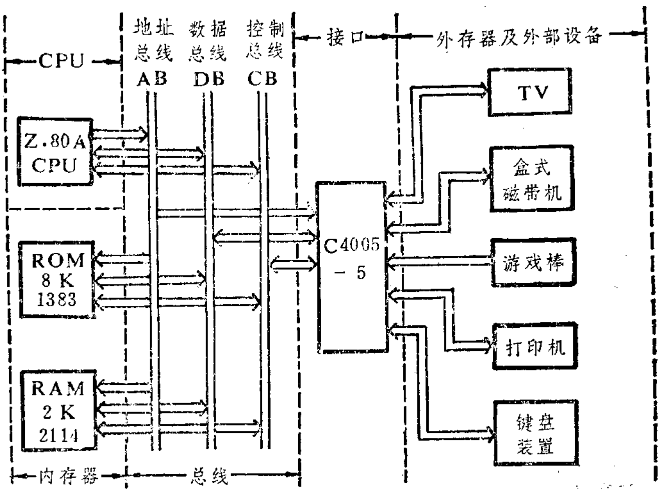


图1-1 R1微型计算机的系统结构

1. 微处理器

R1微型计算机所选用的微处理器，其型号 为Z80A。它是一种 8 位40条引脚的内部单总线结构的微处理器。时钟频率为 4MHz。时钟周期则为$\frac{1}{4}$微秒，执行加法指令的最短时间为 1 微秒。其内部通用寄存器有两组，每组有 8 个 8 位寄存器（包括累加器），可用来寄存常用数组和中间结果。当两个寄存器连起来使用时，便形成寄存器对，可用来存放16位数据或地址码。

2. 存贮器

存贮器分内存贮器（简称内存）和外存贮器(简称外存)两种。在微型计算机中，又把内存 分 为 随 机 存 取 存 贮 器(RAM)和只读存贮器（ROM）。RAM可以随意写入或读出信息（程序和数据），ROM 则 只 能读出事先存贮在它里面的信息，而不能在计算机工作过程中再写入新的信息。RAM是易失型存贮器，当电源关断后，存放在其中的信息也就全部丢失了；而ROM 是固定存贮器，信息存入后就永久固定，不会因断电或操作故障而有所改变。

R1微型计算机的随机存贮器RAM 的型号为2114，容量为2K。它如果与扩展板连用时，则其容量可以扩展到16K或32K。只读存贮器ROM的型号为1383，容量为 8 K，其内装有BASIC解析程序。

R1微型机的外存贮器只有盒式磁带。由于RAM只 能 保存当前正在工作的程序和数据，而当关断电 源 后，存 放 在RAM 中的信息便全部丢失了，因此我们可以把程 序和数据记录在盒式磁带上，在需要使用时再把这些程序和数据调入到RAM 中去执行运算。

3．输入输出接口电路

R1微型机输入、输出接口电路的型号是C4005-5。它的主要功能是实现主机与外部设备之间的连接，通过它可以直接连接键盘、监视器（或家用电视机）、打印机、游戏棒和盒式磁带机等。

4．总线结构

R1微型机的总线：数据总线为8位，可同时传送8位数据，属于双向传送；地址总线为16位，单向传输；控制总线为8位。

R1 微机的总线为单总线结构，这种结构的特点是：①内存器与外部设备相互之间的信息交换，都是通过这组总线来实现的，而不必通过微处理器来进行；②微处理器对外部设备的控制，可以象控制内存器那样来处理；③机内各个部分都是“挂”在单总线上的，它使计算机系统具有积木式的结构形式，如果再配置扩展接口器的话，就可以扩充随机存贮器RAM和多接一些外部设备。

第二节　R1微型计算机的技术性能

电子计算机技术性能的好坏，涉及到很多方面，它不仅仅由主机的性能决定，而且与它的系统结构、指令系统、外部设备及软件的配备等因素有关。所以不能只根据一两项技术指标，就得出计算机性能好坏的结论。例如用每秒运算50万次和100万次的计算机解同一个题目，如果后者的综合性能不如前者的话，则解题时间就可能比前者还要长。因此，只有综合各项指标，才能正确评价计算机性能的好坏。

电子计算机的主要技术指标包括如下几个方面：

1．字长

字长是指计算机字的位数，也就是计算机中运算器所能并行处理的最大二进制数的位数。就以字长来说，大型机一般为32～64位，中型机多为32位，小型机一般为12～32位，微型机为4～16位，目前正向着32位发展。

一般情况下，计算机能处理的字越长，精度便越高，处理能力也就越强。

通常把八位二进制数称为一个字节。16位计算机的一个字长含有两个字节，而8位机的一个字长则为1个字节。在数据处理中，广泛采用ASCII代码（美国标准信息交换码），它的一个字符占用1个字节。

字长亦可作为初步估计计算机数据处理能力和应用范围的依据之一。例如，用字长为8位的计算机进行32位字长的加法运算，需要运算4次，而且还要处理每次运算的进位，既费时又费事。若用字长为32位的计算机进行这种运算，则一次就可以解决问题。所以对复杂问题的运算，宜采用位数长些的计算机。

R1微型计算机的字长为8位。

2. 内存贮器容量

内存贮器容量可以按字长为单位来计算，也可以按字节为单位来计算。每1024个字节，称为1K字节。

一般大型计算机的存贮器容量可达10兆字节，中型机一般为几百K字节至1兆字节，小型机一般为6K至256K字节，微型机一般小于64K字节。

存贮器容量越大，其存贮的数据和程序也越多，因而就能处理越复杂的问题。为了弥补内存贮器容量的不足，可在机外设置外存贮器，如盒式磁带和磁盘等。

R1微型计算机的内存贮器容量是，ROM为8K字节，基

本RAM为2K字节，外接扩展板后可扩展至16K或32K字节。外存贮器只采用盒式磁带。

3．运算速度

由于执行不同的指令所需要的时间不一样，故计算速度的方法亦有所不同。通常采用指令执行的平均时间来计算，也有以执行加法指令所需时间作为标准来计算运算速度的，即运算速度为每秒内执行加法运算的次数，单位为次/秒。大型计算机的运算速度约为十分之几微秒，而微型机则与它采用的微处理器和时钟脉冲频率有关，一般约为1至2微秒。

R1微型计算机的时钟脉冲重复频率为4MHz。运算速度（按执行加法指令所需的最短时间计算）为1微秒。

4．I/O数据最高传送速率

I/O传送速率不仅决定计算机采用什么外围设备，而且也决定它与外界交换数据的速度。大型计算机的传送速率高，可使用高速外围设备，如快速磁盘，快速打印机、快速通信传输线等。而微型计算机的传送速率较低，故常用低速、简单的外围设备，如键盘、显示器、电传打字机等。对参数变化缓慢的变量（如温度、压力等）的控制，则以采用微型计算机较为合适。

5．通用寄存器数目

通用寄存器数目是指微处理器CPU内部的通用寄存器的个数。它是用来寄存常用数据和中间结果的。通用寄存器数目越多，则CPU访问内存器所花费的时间就越少。大型计算机设有的通用寄存器一般有几十个，而微型计算机一般为8个。

R1微型计算机的通用寄存器有两组，每组有8个8位寄存器。

6．软件的配备

硬件是指实际的机器系统，它包括主机及外围设备。而软件则为各种程序的总称。它包括算法语言的解释程序、编译程序及其它程序如：汇编程序，诊断维修系统，故障处理系统，标准程序库，操作系统等。

计算机配备的软件越丰富，则其功效就越显著。大型计算机因具有较大的内存容量和各种外围设备，所以它配有几种高级操作系统，几种高级语言和丰富的应用软件库（如工程计算、商业计算，文件管理，信息检索，声处理等软件）。而简单的微型计算机（例如R1机）一般只提供很少的操作系统，一种高级语言（如BASIC）及简单的用户程序（如标准函数）和解析程序（或编译程序）。目前高档的微型计算机也配有多种高级语言了。

第三节　机器语言和算法语言

计算机是按照人的意志工作的，是由人来指挥的。人如何指挥计算机呢？首先在人和计算机之间要有交流信息的工具——语言。但是计算机并不懂人的语言，无论是中文、英文或科技上常用的数学语言，例如，Z＝X＊Y，2＋3＝5等问题，它都不能识别。因为计算机只能识别“0”和“1”两种信息，所以我们必须将X，Y，Z和数据2，3，5以及运算符号，都用二进制数“0”和“1”组成的代码来表示，才能被机器所识别。

一、机器语言

什么叫机器语言？简单地说，用二进制数“1”和“0”组成代码所表示的计算机的信息（或称指令），称为机器语

言。用机器语言编写的程序，称为机器语言程序或目标程序。由于所编出的程序全是1和0的数字，非常烦琐。不仅存在着难记、难写、难检查等弊病，而且没有通用性，即不同的机器有不同的机器语言，用甲型机器的指令系统编写的程序，不能在乙型机器上使用，这显然是很不方便的，因而给计算机的推广应用带来很大困难。为了解决机器语言的缺陷，人们创造了算法语言。

二、算法语言

算法语言不仅是一种接近于人们习惯使用的语言和通用的数学表示方法的语言，而且容易被机器所接受。

例如，欲计算3ln5之值，如果用机器语言编写程序，就要把乘法运算、求对数值的运算均化为加法运算，因此它包含很多条指令，而且都是用二进制代码表示的。而用BASIC语言编写算题程序时，只需写出PRINT 3*LOG(5)就行了，计算机便能自动计算并打印出结果值。因此，算法语言容易被用户掌握。用户可以不必知道计算机的内部结构、工作原理和指令系统，就可以使用算法语言编写程序，操纵计算机进行各种运算。

用算法语言编写的程序称为源程序。

值得注意的是：我们说用算法语言编写的源程序容易被计算机所接受的意思，并不等于说它能被计算机所识别。人们还必须将用算法语言编写的源程序“翻译”为机器语言程序，计算机才能运转执行。而翻译工作又是由人们事先编好的程序来完成的，这种程序称为翻译程序。

翻译程序有两种基本的工作方式：一种是编译方式，它是将源程序全部翻译为机器语言程序后再加以执行；另一种是解析方式，它是把源程序逐句地翻译，译出一句就立即执

行一句，即边译边执行。一般说来，采用解析方式少占用内存单元，但多花费机器时间。

目前世界上的算法语言有几百种之多，并且还在不断地发展。但比较通用的有如下几种：

（1）FORTRAN语言：它是一种比较适合进行科学计算的语言。

（2）COBOL语言：它是一种比较适用于商业和经济管理方面的语言。

（3）PASCAL语言：它是七十年代最有影响和最重要的语言之一。它是由ALGOL语言发展起来的，用于书写系统程序的一种计算机语言，故又被称为系统程序设计语言。

（4）BASIC语言：它是目前国际通用的计算机算法语言。由于这种语言具有比较节省内存和简单易懂等特点，对内存较少的微型机特别适用，故绝大多数的微型计算机都配有BASIC语言。

第二章　BASIC语言程序的构成及上机操作方法

第一节　BASIC语言的特点及构成规则

一、BASIC语言的基本特点

BASIC—词是英语Beginner's All-purpose Symbolic Instruction Code的缩写，其中文含义为初学者通用符号、指令和代码。凡初学使用微型计算机的人，大都是从学习和使用BASIC语言入手的。学会了BASIC语言，再学其它语言(例如FORTRAN、COBOL、PASCAL等)也就不难了。

本书主要介绍R1微型计算机使用的BASIC语言，但它所使用的语句，对其它类型的微型计算机的BASIC语言同样适用，只是在某些具体规定和个别语句的格式方面稍有差异而已。

BASIC语言具有如下几个特点：

1．简单直观

在BASIC语言中所使用的词、符号，和英语中使用的词及数学上使用的符号差不多，因而BASIC语言是一种容易理解、记忆和易学易懂的语言。

2．具有人机对话性能

由于采用BASIC语言，使用者通过键盘，计算机通过显示器，就能实现人和机之间的对话。对于R1微型机的BASIC语言，当使用者通过键盘输入源程序时，每打入一个程序行，计算机立即检查键入的内容有无错误。如果有错的话，

它会拒绝接受程序，并通过显示器显示出错误的位置，待使用者将程序修改以后，才允许输入。此外，R1微型机还能将键盘打入的内容和已输入到内存的程序行，分别显示在显示屏的不同位置上，以使人机对话更加直观。

在执行程序过程中，计算机又能按照执行的顺序去检查程序，并把执行情况向使用者报告。如遇到错误之处，便指出错误的性质和位置。这时使用者可以再通过键盘修改程序，在执行修改后的程序时，计算机又能继续检查程序。因此使用BASIC 语言编程，可以在上机操作时边试边改，直至得到满意的结果为止。

3．允许在键盘上直接进行运算。

如同使用计算器一样，在计算机上使用BASIC 语言时，可以通过键盘直接进行各种算题的运算，而不必编写程序。

4．具有一定数据处理功能和字符串处理功能。

由于BASIC 语言具有这两种功能，使它能够方便地用在小型数据管理和事务管理方面。

二、BASIC语言程序的构成和基本规则

先举一个简单的例子，说明什么是 BASIC 语言程序以及程序构成的基本规则。

【例 1 】求某班某门课程的平均成绩。

已知：5 分人数A = 13人

4 分人数B = 12人

3 分人数C = 5 人

2 分人数D = 2 人

全班人数U = 32人

平均成绩的计算公式为

$$G = (5A + 4B + 3C + 2D) \div U$$

计算本题时，可先用 BASIC 语言编出下列程序：

```
10  LET  A=13
20  LET  B=12
30  LET  C=5
40  LET  D=2
50  LET  U=A+B+C+D
60  LET  G=(5*A+4*B+3*C+2*D)/U
70  PRINT  G
80  END
```

这种利用BASIC语言编制的程序，称为BASIC 语言程序，也称为BASIC语言源程序。从上面的程序，可以分析、归纳出BASIC语言程序的构成具有下列基本规则：

1．程序是由一系列语句组成的

本例的程序由 8 个语句组成。基本 BASIC 语言程序规定一行写一个语句，而扩展 BASIC 程序，则允许一行写几个语句，还规定两个语句之间必须用冒号“：”隔开。例如：

```
10  LET  A=13：LET  B=12
10  A=13：B=12
```

（扩展 BASIC 中的LET 可省略不写）

R1微型计算机的BASIC程序，其每一行只能写一个语句，但可以将LET省略。

通常，每一个语句具有一项功能。例如，10行语句的功能是把数值13赋值给变量A，70行语句的功能是把 G值显示在显示屏上。

2．每一个语句一般由标号、定义符和语句体三个部分组成。

（1）标号（或行号）。每个语句开头的数字称为行的

标号，或简称为行号。例如上面程序中的10，20等都是行号，它们所在的程序行分别简称为10行、20行。

标号是一个无符号的整数，其范围是0～99999，它的上限值随计算机型号不同而有所不同。例如，R1微型计算机的标号范围是0～9999，而TRS-80微型计算机的标号范围是0～65529。

编写程序时，标号可以不连续编号。一般标号的间隔为10，以便在调试程序时，能在行与行之间补充新的程序行。计算机在执行程序时，按标号由小到大的顺序执行，而不是按输入程序的先后次序执行。例如，输入程序先后次序的标号是10→20→30，在修改程序时添加了一行语句，其标号是15，显然第15行语句是最后输入的，但就在输入此程序时，微型机的BASIC解析系统，会自动将标号按由小到大的顺序整理排列好，把15行插在10行与20行之间，并且在执行程序时，按标号大小顺序执行，即执行10行以后马上转到执行15行，然后再执行20行。这也就说明了为什么在开始编程序时，标号不连续编号而留有一定间隔的原因。

（2）语句的定义符。每一个语句的行号后面，都有一个英文词用来表示语句的类型，以规定计算机执行某一特定的功能，这个英文词称为语句的定义符。例如上述程序中10行的LET，就是这个语句的定义符，它规定计算机执行"赋值"功能，即把13这个值赋给变量A。

（3）语句体。跟在语句的定义符后面的部分就是语句体，它用来说明该语句所要执行的具体内容。例如，虽然10行和50行同属于赋值语句，但它们的具体内容不同，10行的内容是将数13赋给变量A，而50行的内容是将A，B，C，D的值相加后，再把它们的和送到变量U中去。

3．每个程序的最后应以END语句结束。

计算机在执行程序时，遇到END语句即停止工作。但在扩展BASIC语言中无此项规定。R1微型计算机没有END语句，必要时可用暂停语句STOP来代替。

4．一个语句必须在一个程序行内写完，如果一行内写不完，其语句体部分不能接着写在第二个程序行内，而只能把一个语句拆成两个或多个语句。

例如：

```
10  PRINT"A…"; "␣␣"; "B…"
```

虽然，语句体"B…"在一程序行内写不完，但不能写成

```
10  PRINT"A…"; "␣␣";
20  "B…"
```

因为20行无定义符，显然它不能构成一个语句。此时，可以把它拆成两个语句来写：

```
10  PRINT"A…"; "␣␣";
20  PRINT"B…"
```

这里要注意的是，程序行不等于书写行和显示行。不要把一个语句必须在一个程序行内写完的规定，错误地理解为一个语句不能在两个以上的书写行中书写，或不能在两个以上的显示行中显示。

究竟一个程序行能容纳多少个字符（指英文字母、数字、空格以及其它有效符号）呢？不同的计算机有不同的规定。例如TRS-80微型机规定其一个程序行的字符最多不能超过255个，显示行为64个字符；而R1微型计算机规定其一个程序行可含有512个字符，显示行为32个字符。这就是说，允许一个程序行的字符在几个显示行中显示。

第二节　R1微型计算机的键盘

一、键盘符号

R1微型计算机的键盘共有42个键，如图2-1所示。

数字键：φ，1，2，…9共十个。请注意，计算机的数字0用φ表示，以便与英文字母O区别开来。但由于书本中的数字0与英文字母O不易混淆，故书写时数字零一般都用0而不用φ。

字母键：A，B，C，D，…Z共26个。只有大写字母，没有小写字母。

标点符号键：，（逗号），·（小数点），；（分号），“　”（引号），（　）（圆括号），SPACE（空格）。

运算符号键：+（加号），-（减号），*（乘号），/（除号）。

比较符号键：=（等号），＜（小于），＞（大于），＜=（不大于）（先按＜键，再按=键），＞=（不小于）（先按＞键，再按=键），＜＞（不等于）（先按＜键，再按＞键）。

逻辑运算符：AND（逻辑和），OR（逻辑或），NOT（逻辑反）。这些运算符没有单独的键，使用时要逐个地按压字母。

其他符号键：$（读作打纳），π键以及22个图样符号键。

各种函数键：SIN（正弦），COS（余弦），TAN（正切），ASN（反正弦），ACS（反余弦），ATN（反正切），LOG（对数），SGN（符号），SQR（平方根），RND（随机），EXP（指数），ABS（绝对值）等。

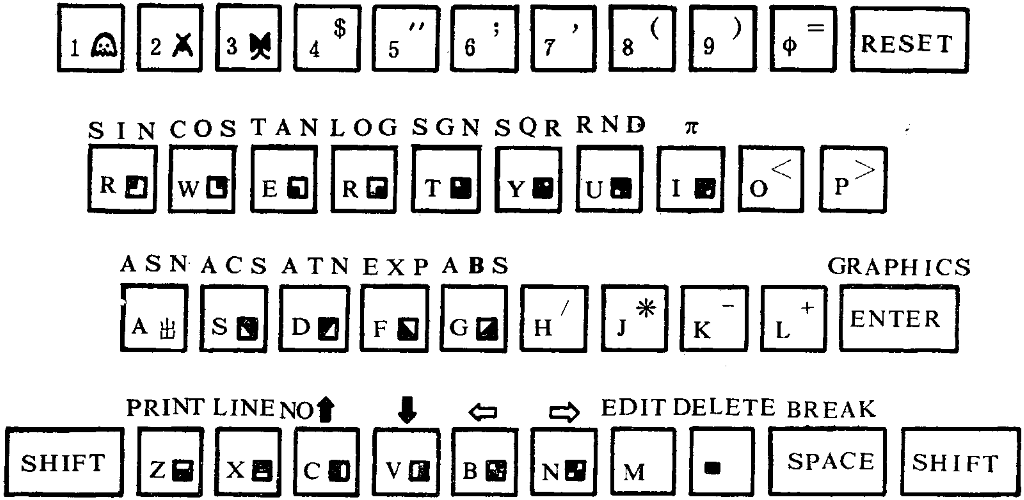


图2-1 R1微型计算机的键盘

二、特殊功能键

按下特殊功能键，就可以命令计算机执行特定的功能。R1机共有下面12个特殊功能键：

存入键(ENTER键)。每打入一个程序行或命令内容后，按下此键，才能把信息输入到计算机的内存中；对于无行号的语句或命令，计算机会立即解析执行。每按下ENTER键，闪动方块光标立即回到起始位置，显示内容立即换行，打印机回车，故ENTER键又称回车键。

键符调整键（SHIFT键）。利用此键，可以使一个键得到两个字符，即相当于一个键等效于两个键使用。如再利用图形键，使计算机进入图形状态后，再来按键，又可以得到另外两种字符。这样一来，一个键便可以等效四个键使用。

现以A键为例（参阅图2-1键盘图）加以说明：

（1）只按A键时，则打入的键符是字符A的本身。

（2）先按住SHIFT键（按住不放），再按一下A键（按下即放），然后再放SHIFT键（即先按后放SHIFT键，后按先放A键。这种操作方法，以下简称同时按SHIFT键和A键），此时得到的键符是A键上方（有些键是指上右方）所示的键符ASN。

图形键(GRAPHICS键）。按此键（同时按SHIFT和ENTER键），可以使计算机进入图形状态；此时光标也由闪动的方块光标变为闪动的G字光标，以便与正常的命令状态区别开来。再按其他键时，显示屏上出现白底黑字的字符（下称反字符）或图案字符。例如进入GRAPHICS状态后，按A键，则得到白底黑字的A（即反字A）字。如果同时按SHIFT键和A键，则显示出A键右方所示的图案字符“出”。

如果要退出GRAPHICS状态，只需按ENTER键就可以

了，此时闪动的G字光标立即消失，并且恢复为闪动的方块光标，计算机又处于正常的等待命令状态。

删除键(DELETE键)。按一次此键（同时按SHIFT键和·键），可删去光标左方的一个字符。连续按几次，则删去几个字符。

左移键(←—键)。按一次此键（同时按SHIFT键和B键），可以使光标退回一格或左移一个字符。连续按几次，则光标退回几格。

右移键（—→键）。按一次此键（同时按SHIFT和N键），可以使光标右移一个字符。连续按几次，则光标右移几个字符。

上移键（↑键）。按一次此键（同时按SHIFT键和C键），可以使程序光标上移一行。连续按几次，则光标上移几行。

下移键（↓键）。按一次此键（同时按SHIFT键和V键），可以使程序光标下移一行。连续按几次，则光标下移几行。

编辑键（EDIT键）。按此键（同时按SHIFT和M键)，可以使计算机进入编辑状态（或删改状态）。这时，计算机将程序光标所指的那行内容显示在屏幕的下部，以便为该程序行的删改做好准备。

中断键（BREAK键）。分两种情况来讨论：①在程序执行中，按此键（同时按SHIFT键和SPACE键），可以中断程序的执行，使计算机脱离执行状态而进入命令状态，一直到打入CONT命令和GOTO（行号）命令后程序才继续执行；②当键入的程序或数据出错时，按下此键则删去该错误行的全部内容。

自动编行号键(LINE NO键)。利用此键可以自动编行号以代替从键盘键入行号。按此键(同时按SHIFT键和X键)时，得到的行号是将程序光标所指的行号加10。例如，在输入新程序时（刚接通电源或打入NEW命令后），程序光标指在0行位置上(没有显示出来)，因此第一次按此键时，得到的行号是10，以后每输入一程序行的开始时按下此键，则得到的行号的顺序依次是20，30，…，即间隔为10。如果程序中已插入了其它行号（例如15），而程序光标指在15行位置上，此时，如按下LINE NO键，则得到的行号为25。

重新启动键（RESET键）。按此键，可以重新启动BASIC解析系统，将程序重现在显示屏幕上。

第三节 上机操作方法

R1微型计算机的最大特点是操作简单，价格低廉。它的主机面板是一键盘，键盘后面有六个插孔(见图2-2)。通过这些插孔，可以将主机和外部设备连接起来。插孔的用途如下：

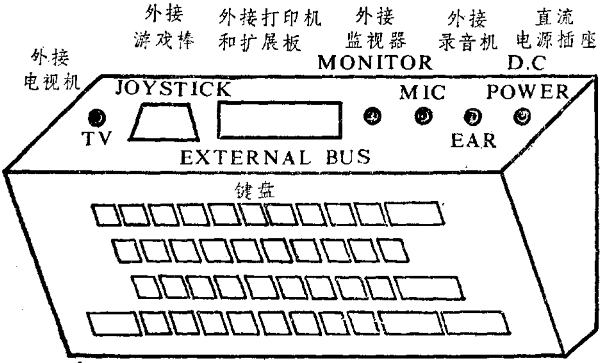


图2-2 主机与外部设备的连接

DC POWER——用来接9伏直流电源；

EAR和MIC——用来接普通录音机；

TV——用来接一般家用电视机；

MONITOR——用来接显示器；

JOYSTICK——用来接游戏机操作杆（游戏棒）；

EXTERNAL BUS——接打印机或扩展存贮板。

一、外接电视机和电源的操作

R1机装有电源变换器，作用是将220伏交流电变换为9伏直流电源。变换器两端各有一根带有插头的连接线，使用时将电源插头插到220伏交流电源上，另一个插头插到标有DC POWER的插孔中。如果不小心将此插头插到其它插孔上，也不会发生什么危险。

R1机上没有电源开关。当插好电源插头后，如果键盘右上角的红色指示灯发亮，同时发出响声，这就表示已接通电源了，计算机就会自动启动BASIC解析程序，而处于BASIC解析程序的管理状态中。但是一定要记住，一旦切断电源，存贮器存贮的全部内容都会丢失，因此在工作时不要轻易断开电源。

计算机如何连接显示器呢？如果用普通电视机作显示器的话，可将与R1机配套的圆形同轴电缆的一端接于电视机的天线输入插孔，另一端接R1机的TV插孔，这时，微型机的信号是通过天线插孔送入显示器的。如果不用电视机，而用监视器作显示器时，则将同轴电缆的一端接显示器的视频输入插孔，另一端接R1机的MONITOR插孔。这时，R1机是直接将视频信号送给显视器的。所以用监视器显示比用普通电视机显示更为清楚些。

如果电视机使用的是平衡馈线，则它的阻抗是300欧姆，

而R1机提供的天线电缆，其阻抗是75欧姆，因此，应使用一个阻抗匹配器，把300欧姆转换为75欧姆，才能使显示的画面清晰稳定。如果电视机和R1机的阻抗不能很好地匹配，则计算机的信息就可能在电视机上显示不出来。

因为计算机工作时，我们不听声音而只看画面，故应将用作显示器的电视机的音量调到最小位置，亮度亦不应太强，频道选择开关可指在VHF的第二频道上。

R1机与电视机的连线接好后，再分别接通它们的电源，R1机右上角的红色指示灯就会发亮，同时其喇叭发出“滴”声。当在电视机显示屏的左下角出现一个闪动的小方块（光标点）时，表示计算机处于命令状态，可以开始工作了。

如果接通电源后，显示屏上不出现闪动的光标，可检查一下电视机的频道选择开关是否指在二频道，同时调节微调旋钮和行同步旋钮，直至出现闪动的光标为止。如果仍然得不到闪动的光标，可按一下R1机上的RESET键，此时显示屏幕的左上角应出现READY（准备好了），而左下角出现闪动的方块光标。如果还是得不到光标，就要对计算机进行冷启动。其做法是将电源插头拔出来再插上，或者先按下ENTER键（按住不释放），再按RESET键，释放时先放RESET键，后放ENTER键，这样就会得到闪动的光标了。

二、上机操作步骤

编好程序后，就可以上机操作，其操作过程大致分三步：清内存、输入程序和执行程序。

1．清内存

在输入一个新的程序之前，必须先清除内存中原有的程序和变量数值。其做法是：当显示屏幕左下角出现闪动的光标时，表示计算机处于命令状态，此时打入NEW命令（通

过键盘依次逐个打入N、E、W三个键，再按ENTER 键），显示屏的左上角出现READY，右下角出现一个闪动方块光标（见图2-3）时，就表示计算机已清除内存中原有的程序，为输入新的程序做好了准备。

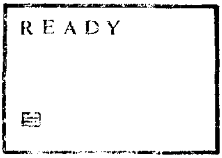


图2-3

如果已知内存中并没有保存任何BASIC程序（例如刚接通电源），则可以省去这一步操作。

2．输入程序

把编好的源程序的每一程序行通过键盘输入时，除键盘上设有专门键的某些字符（如各种标准函数键和 PRINT 键等）可直接按键外，其他内容均应逐个打入字符。这里再强调一下，每打完一个程序行，都要按一下 ENTER键，告诉计算机该行信息已输入完毕。计算机收到 ENTER 键信号后，就把该程序行的内容保存在内存中，并在显示屏上显示出来，同时将显示屏下面的闪动光标自动移到起始位置。

在操作过程中，每打入一个键符，都会在显示屏下面显示出来，此时闪动的方块光标亦相应右移一格，指出将要显示的字符位置。根据显示内容，便可以及时地检查打入的内容有无差错。

R1微型机有两个光标，一个是上述的闪动方块光标，其作用是表示计算机处于命令状态，和指出即将键入字符的显示位置；另一个是称为程序光标的白底黑字字符“图”，它出现在显示行标号的右侧，表示这一程序行是刚才键入的。

3．执行程序

上述步骤只是将程序输入计算机，而计算机并没有执行这些程序。当程序输入完毕后，应打入RUN命令（从键盘

上逐个打入R、U、N三个键，再按ENTER键），计算机才会自动执行存在内存中的BASIC程序。一般是按照行号由小到大，同一程序行则由左到右的顺序去执行程序的，只有在遇到改变执行顺序的语句时，才改变上述执行顺序。

执行结果在显示屏上显示出来。

三、上机操作实例

例如：

```
10  X = 5
20  PRINT  "X ="; X
```

虽然这是一个很简单的程序，但它仍然可以反映出上机操作的全过程，操作步骤如下。

1．清内存

逐个打入N、E、W键，再按ENTER键，显示屏上画面如图2－3所示。

2．输入程序

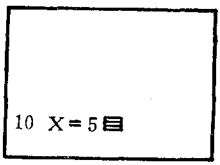


图2-4

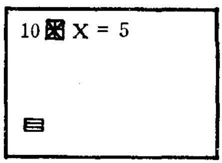


图2-5

（1）用键盘打入程序行第10行的内容。按键次序是：[1] → [0] → [X] → [=]（SHIFT键和 φ 键）→ [5]，每按下一键，光标右移一格；打入第10行内容后，画面如图2－4所示。

按ENTER键后，画面如图2－5所示。

（2）用键盘打入第20行的内容，按键次序是：

[2] → [0] → [PRINT] → ["] 键→ [X] → [=] → ["] 键→ [;] 键→ [X]。打完程序行内容后，按ENTER键，此时画面如图 2－6 所示。

3．执行程序

逐个打入R、U、N键后，再按ENTER键，在显示屏屏幕的上部显示出程序运行的结果，如图 2－7 所示。同时在屏幕的左下角显示 OK IN 20，表示“对了，程序执行到20行”。

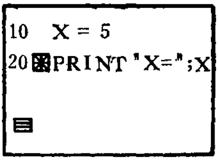


图2-6

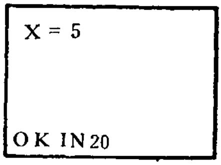


图2-7

第四节 键 盘 命 令

一、键盘命令与特殊功能键的区别

在上一节的上机操作过程中，清内存和执行程序都是由使用者通过键盘打入N、E、W和R、U、N命令后，再按下ENTER键而实现的。这种通过键盘打入的命令，称为键盘命令。而特殊功能键是通过按键后产生作用的。虽然两者都能完成某一特定的功能，但它们之间是有区别的。打入键盘命令后，必须按一下ENTER键，计算机才能接受命令；而按下特殊功能键后，则不需再按ENTER键，计算机就可以立即执行该键的功能。另一个区别是在打入键盘命令时，显示屏的下方会显示出它们的键符；而按下特殊功能键后，它的键符不被显示出来。此外，所有键盘命令都具有先清除屏

幕上的画面，再执行命令中内容的功能；而特殊功能键则不一定有清画面的功能。

二、R1微型计算机的键盘命令

每一种BASIC语言的解析系统，都配有一套功能很强的、结构完整的键盘操作。这种键盘操作分为两类：键盘命令和键盘运算。后者将在本章第九节讨论，本节只对常用的一些键盘命令作简单的介绍。

1．NEW（新的）

它用来清除计算机内存中所存贮的BASIC源程序及变量，为接受新程序做好准备。

2．RUN（运行）

它用来命令计算机执行内存中的BASIC程序。若打入RUN命令后接着打入行号n，则从n行开始执行：

打入RUN，则计算机从最小行号开始执行。

若打入RUN　100，则计算机从100行开始执行。

值得注意的是，RUN命令执行完毕后，所有程序仍然保存在内存中。这可以用重复打入RUN命令后，观察是否得到同样的结果来证明。但如果打入的是RUN　n命令，则会清除小于n行的程序中所赋给变量的值。例如：

```
10  X=5
20  PRINT  X
30  Y=10
40  PRINT  Y
RUN  ENTER
 5
10
RUN  ENTER
```

5

10

RUN 40 ENTER

这时，在显示屏下方立即出现UV IN 40的报告信息，它指出40行有未定义变量的错误。因为RUN 40命令把变量X和Y的值都清除了，因此执行40行的内容时，变量Y便没有定义了。此时如果打入LIST命令，显示屏上仍然重现原有程序，说明RUN n命令清变量而不清程序。

3．LIST（列程序清单）

它用来命令计算机输出显示内存中所有源程序的清单。注意：此时程序光标没有出现，实际上光标是指在行号为0的位置上。

LIST n（列出从n行开始的程序清单）。打入LIST n命令后，屏幕上显示n行及大于n行的程序清单，此时程序光标指在n行位置上。R1微型计算机上的这条命令的功能，与某些计算机有所区别。例如，在TRS-80微型计算机上，打入LIST n命令后，只显示n行的内容，而打入LIST n-后，才显示出n行及大于n行的程序清单。

由于R1微型机只有22行显示行，当程序多于22行时，就要分段列出程序清单。例如，某一程序共含有40行，行号由10开始，间隔都是10，那么，这一程序的清单，就要分两段打入：①先打入LIST命令，列出10行至220行的程序清单；②再打入LIST 230命令，才接着显示230行至400行的程序清单。

4．CLS（清屏幕）

它用来清除显示屏上的画面，但不清除内存中所存贮的内容。

CLS命令可以作为程序中的一个语句来使用，即可以在CLS前面加上一个行号，如100　CLS。这样，当程序执行到100行时，便清除了显示屏上的画面。

5．CLEAR（清变量）

清除内存中所有变量值，并空出它们所占的内存空间。例如：

```
10  A=10
20  PRINT  A
RUN  ENTER
10
打入CLEAR  ENTER
接着打入PRINT A  ENTER
```

这时，在显示屏的下方将出现UV字样，它说明打入CLEAR命令后，清除了变量A的值。如果此时打入LIST命令，则又会重现原有程序清单，说明CLEAR命令只清除变量值，而不清除内存中的源程序。

6．CONT（继续）

它用来命令计算机继续执行程序。CONT在下列两种情况下使用：①当计算机在执行程序时，由于按下BREAK键而中断了程序的执行，并给出一个报告信息“BK IN（行号）”，表示程序中断在所示的行号处，此时如打入CONT命令，则程序将从报告信息中所指的行号继续执行；②在计算机使用STOP语句而暂停程序的执行，并给出一个报告信息ST IN（行号）时，如果打入CONT命令，则程序将从STOP语句的行号的下一行继续执行。

如果暂停期间改动了程序，则CONT命令无效，此时只能使用RUN命令。

7. n　ENTER（删除n行）

这个命令是在打入行号n后，再按ENTER键，便将行号为n的程序行的内容删掉。

上述的键盘命令，除CONT命令外，其余命令都允许以语句的形式放在程序中使用，与前述100　CLS的情况相类似。

除了上面介绍的几种常用的键盘命令外，还有其它键盘命令，它们将分别在有关章节中讨论。

第五节　源程序的删改

一、什么叫程序的删改?

当我们把编好的程序进行上机操作时，经常会出现如下情况：在输入程序的过程中，难免会发生错按、漏按和多按字符的操作错误，或对程序本身考虑不周，或语法有错误等。如发生上述情况，便要求在操作过程中修改或补充程序。归纳起来，程序的删改有如下工作：

（1）增加某些字符或语句；

（2）删去某些字符或语句；

（3）改正某些字符或语句。

我们把上述工作称为程序的删改或编辑。如果利用屏幕进行程序修改，就称为屏幕编辑。

要修改某一程序行的内容，最简单的方法是把正确的程序行的内容重新输入一遍，而使用的行号要与欲修改的行号一致。

例如，原来错误的程序行为

```
10　PRINT　X=5
```

要改为

10 PRINT X

操作方法是，打入正确的程序行内容后，再按下ENTER键。但是，如果修改的程序行很长的话，采用这种修改方法就显得费时费事，而且重新打入字符时也很难保证不会出错，因此这不是一种好方法。

每种机的BASIC系统都配有一套完整的编辑功能，但不同机型有着不同的操作方法，因此，操作前必需熟悉和掌握所用计算机的删改功能。

二、R1微型机的字符删改操作方法

与其他微型计算机比较，R1微型机在删改功能方面具有操作特别简单，直观和容易掌握的特点。现将R1机的删改操作方法分述如下。例如已编好如下程序：

```
10 X=3
20 Y=6
30 Z=9
40 S=X+Y+Z
50 PRINT "X="X; "Y="Y; "Z="Z
60 PRINT "S="; S
```

在输入程序时，可试做如下删改工作：

1．删字符

设打入第10行程序时，在未按ENTER 键之前，发现在字符 3 的前面多打了一个 0 字符，屏幕下方显示出

10 X=03▤

为了删去 0 字符，可以进行如下操作：

（1）按一次⇦键，将光标移至要删改字符的右边一格位置上，这里是将光标左移一格，屏幕下方出现

10 X=0▤3

（2）按一次DELETE 键，删去光标左边一个字符；

（3）按ENTER键，把删改后的10行内容X＝3存入内存中。

2．增字符

打入上例程序行50行的内容，按下ENTER 键后，屏幕下方出现：

50　PRINT　"X＝"?X；"Y＝"Y；"Z＝"Z

程序中出现出错符号?，就说明键入的信息有错误，这时还伴有一个声音，以提醒用户注意，已经出错了。出错之处一般是在?所指示的位置，或者在其左边的字符位置。

仔细检查一下，就会发现这是属于语句上的错误，在?所指示的位置处漏打入"；"分号（理由参阅下章PRINT语句），这就需要增加字符。

增加字符的操作方法很简单，只要按下所漏打的分号字符就可以了。如果不是在?处漏打字符，就利用左移或右移键将光标移至漏打之处，再打入漏打字符。如本例后两项内容Y和Z的左方也漏打了分号字符，按照上述方法增加字符即可。

修改后，50行的正确内容为

50　PRINT　"X＝"；X；"Y＝"；Y；"Z＝"；Z

3．改字符

设在输入上列程序60行内容时，按S字符键前忘记释放SHIFT键，变为错打入ACS字符，这时屏幕下方出现

60　PRINT　"ACS＝"；S▤

现在需要删去ACS字符，改正为S字符。

操作方法相当于上述两种内容的综合。即

（1）按四次左移键←，将光标左移4格，移至要删改字

符ACS的右边一格位置；

（2）按一次DELETE键，删去ACS字符；

（3）按S键，补入S字符。

三、R1微型机的语句删改方法

1．改语句

设要将上列程序中的50行语句改为

50 PRINT "X="; X; "␣␣"; "Y="; Y; "␣␣"; "Z="; Z

操作方法如下：

（1）按上移键↑，将程序光标移至行号50的右边；

（2）按EDIT键，将程序50行内容移至屏幕下方；

（3）用上述删改字符的方法改正之。即：按8次右移键，将光标右移至"Y="的左边，再打入"␣␣"；字符。用同样的方法再修改Y；后面的"␣␣"；字符。

2．删去一行内容

利用n ENTER命令，可删去n程序行。例如打入60按ENTER键后，即删去60行内容。

3．增加一程序行内容

打入新增加的程序行内容后，按ENTER键，即将新增加的程序行存入内存中，并按它的行号大小排列在程序之中。

例如，打入 70 PRINT S/3(平均值)后，按ENTER键，就把这行内容排列在60行的后面。

综上所述，R1机删改程序的方法可归纳如下：

（1）利用上移或下移键，将程序光标移至待修改程序行的行号的右边位置上。

（2）按下EDIT键，使待修改的程序行内容呈现在屏幕下方。把它看作是刚打入的程序行。

（3）利用左移键或右移键，将光标移至待修改字符的右边。

（4）利用DELETE键，删去光标左面的字符。连续按几次，可以依次删去几个字符。

（5）删去错误字符后，再打入所需要的正确字符。

第六节 BASIC语言中数的表示方法

BASIC语言中数的表示方法与日常生活中数的表示法差不多，也是采用十进制形式，不过有一些具体规定。此外，对非常大的数和非常小的数，要用指数记数法来表示。

一、日常记数法

1. 采用十进制形式计算，但规定一个数内不能用逗号分位。例如，1000000不能写成1,000,000，因为这两种写法用在打印语句时，其打印输出的结果是不一样的。

2. 数可以是正数，也可以是负数，但规定正数的正号“+”省略不写。正数若写上“+”号反而被认为是错误的，而负数的负号“-”则不能省略。

3. 小数点前面的0和最后面的0可以省略不写，但写上去也可以。如0.12000也可以写成.12，打印出的结果都是0.12。

4. 数值范围

不同型号的计算机各有不同的数值范围。R1微型计算机能处理的数X的范围为$9\times10^{-39}\leqslant|X|\leqslant1\times10^{38}$。用数轴表示如图2-8。

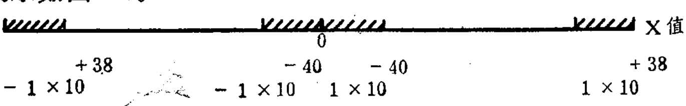


图2-8

有斜线之处表示在数值范围之外，即R1微型机不能处理 -1×10^{-40} 至 1×10^{-40} 之间的数和大于 $1\times10^{+38}$ 以及小于 $-1\times10^{+38}$ 的数。它把小于|X|下限的数当作零处理；当出现大于|X|上限的数时，则在屏幕的左下角显出溢出信息OV（Overflow）。

5. 数的有效位

计算机能处理的数据的有效位是有限的。R1微型机输出数字的有效位为8位。如果键入的数字是1234567.89（共九位）时，R1微型机便自动进行四舍五入处理，输出显示结果只有八位有效数，得到1234567.90。对于整数，它也是作同样的处理。例如：键入123456789（九位数字）时，则只显示出123456790；当键入1234567890123（十三位数字）时，则也只显示出1234567900000。

从上述例子可知，R1微型机输出数字的有效位都是8位。当输入14位以上的数字时，输出结果则用指数记数法表示。

二、指数记数法

指数记数法又称为科学记数法或E记数法。

对非常大的数或非常小的数，可以用指数形式来书写。例如，$1000000000=1\times10^{9}$，$0.0000000001=1\times10^{-10}$。这种表达数字的方法称为指数记数法。由于在BASIC语言中，10的右上角的幂次无法表示，所以用E代表乘方的底数10，幂次亦用同样大小的字符在同一行书写（即无上角、下角标记）。如：

表示的数	输出形式
1×10^{7}	1E+7（1不能省略）
1×10^{9}	1E+9（指数中“+”号不能省）
-1×10^{-9}	-1E-9

-9.876×10^{-9}　$-9.876E-9$

E的前面的数称为数字部分或称为尾数，其正号可省略不写。E后面的数称为阶码，阶码中的符号不论是正号还是负号均不能省略。

第七节　BASIC语言中的标准函数

用BASIC语言规定的一些函数叫做标准函数。用户使用标准函数时，在源程序中只需写出函数名和自变量就可以直接引用，而不必专门写出一段计算这些函数的程序。因为这些标准函数已被编成子程序，放在只读存贮器ROM中。

标准函数中的自变量，可以选一个具体的数值，也可以用表达式来表示，通常要用括号括起来。R1微型机容许不加括号，如SIN π。但当自变量是一个表达式时，就必须用括号将它括起来，例如SIN(3+A)。

R1微型机中的BASIC系统，共有十二种标准函数。下面分五类介绍这些标准函数。

一、三角函数

BASIC语言中的三角函数，包括正弦函数SIN(X)、余弦函数COS(X)和正切函数TAN(X)三种。函数中自变量X的值必须以弧度为单位。例如

sin1°应写成SIN(1*π/180)

cos30°应写成COS(30*π/180)

tg45°应写成TAN(45*π/180)

此外，正切函数中自变量X值只能接近π/2或π/2的奇数倍。如果等于±π/2，则函数值将超出R1微型机允许的数值范围。

二、反三角函数

BASIC语言的反三角函数包括ASN(X)反正弦函数、ACS(X)反余弦函数和ATN(X)反正切函数三种，计算结果的函数值的单位也是弧度，而且ATN(X)值只能在$-\pi/2$至$+\pi/2$范围内变化。

三、指数、对数和平方根函数

1．指数函数EXP(X)

它用来计算e^x之值，其中$e=2.71828\cdots$。EXP就是代表以e为底的指数。它不能简写为E，否则，便与数的指数记数法混淆在一起了。自变量X可以是一个常数值，也可以是一个简单变量或者是算术表达式，但它们之值均不能使该函数值超过R1微型机的允许范围。

例如，e^7应写成EXP7，不能写成E+7或E＊＊7。

2．对数函数LOG(X)

LOG(X)是以e为底的自然对数，不是代表以10为底的对数。如果要计算以10为底的对数值，可利用对数换底公式来进行换算。即$\log_{10}x=\ln x/\ln 10$。例如求$\log_{10}500$之值，其BASIC函数式为LOG500/LOG10。用PRINT LOG500/LOG10可进行直接运算：按ENTER键，即能得到2.69897的结果值。

我们知道，$\ln e=1$。如果打入PRINT LOG2.7182818，则得到0.99999999的结果值。因为2.7182818不是真正e值，而只是e的近似值，所以求得的结果也不是1。

3．平方根函数SQR(X)

此函数用来计算X的平方根。如打入PRINT SQR 9，按ENTER键后，可得结果值为3。这个函数也可写成X＊＊0.5，但速度比直接用SQR(X)慢些。应用此函数时，要注

意X值不能为负值，否则作为错误处理。

四、绝对值函数和符号函数

1．绝对值函数ABS(X)

它的功能与初等代数中求|X|的值一样。例如：PRINT ABS 5和PRINT ABS(－5)的结果值都是5。

2．符号函数SGN(X)

SGN(X)是用来求X的符号值的。当X为正数时，SGN(X)＝1；当X为负数时，SGN(X)＝－1；当X为0时，SGN(X)＝0。例如

```
10  X=5
20  Y=-5
30  Z=0
40  PRINT  SGN(X)
50  PRINT  SGN(Y)
60  PRINT  SGN(Z)
RUN
 1
-1
 0
```

五、随机函数RND(X)

RND(X)随机函数是用来产生模拟随机量的。它的有关内容将在第九章讨论。

第八节　变量、运算符、运算规则和表达式

一、变量

在BASIC语言中有两类变量：数值变量和字符串变量。

数值变量又分为简单变量和下标变量两种。关于下标变量和字符串变量，将分别在第七章和第八章中专门讨论。

简单变量通常用英文26个字母中的任一个来表示（如A，B，C，X，Y，Z等），或者用一个英文字母后面跟着一个0至9的数字来表示（如A1，B2，C3等）。但是，不容许用前面是数字后面用英文字母（如1A，2B等）作为变量名字。

BASICⅡ语言程序规定（如TRS-80微型机）：简单变量名可以用一个字母后跟一个数字或一个字母来表示。简单变量名可以多于两个字符，但计算机只能识别开头两个字符，即有效字符只有两个。

R1微型机的BASIC程序对简单变量名没有上述两种限制，只要第1个是英文字母，后面跟着写几个数字或几个字母都可以。这样一来，可以直接取用某些英文单词如PHYSICS（物理）、ENGLISH（英语）等名词作为变量的名字，使阅读程序时对变量值的含义一目了然了。

其它的符号如α,β,γ,δ,ε等，均不能作为简单变量名字。

二、运算符

1．算术运算符

算术运算符有：+（加），-（减），*（乘），/（除），**（乘方）共五种符号。某些计算机则用符号↑代表乘方。

注意：在表达式中*号不能省略，也不能用“·”来代替。例如，不能把A*3写成A3，6*8写成6·8，因为这些写法分别与简单变量A3、数6.8相混淆。

2．比较关系符

比较关系符有：=（等于），＞（大于），＜（小于），＞=（不小于），＜=（不大于），＜＞（不等于）共六种

符号。

3. 逻辑运算符

逻辑运算符有：AND（与），OR（或），NOT（非）共三种符号。

三、运算规则

表达式的运算，必须按照下面顺序进行。

（1）先乘除，后加减（＊、/优先于+、-）。

（2）乘方优先于乘、除。

（3）函数优先于乘方。

（4）括号最优先。在BASIC语言程序中，括号无大、中、小之分，一律用圆括号（ ），可以在（ ）内套用（ ）。

用箭头表示运算的先后次序如下：

（ ）⟶函数⟶乘方⟶＊、/⟶+、-

```
例如： 2 * SIN ( 3 * X ) * * 5 / 7 + 8
        ↑  ↑        ↑        ↑   ↑   ↑
        ④  ②        ①        ③   ⑤   ⑥
```

小圆圈内的数字表示运算的顺序，也就是说，先进行括号内的运算，再进行函数SIN的运算，求出SIN（3 ＊ X）后才进行自乘5次方的运算，乘和除是相同等级的，按先左后右次序运算，最后是加8。因此上式代表的代数式是：$\frac{2\sin^5 3x}{7}+8$。如果对优先次序没有弄清楚的话，上式容易被误解为$2(\sin 3x)^{\frac{5}{7}}+8$或$\frac{2\sin 3x^5}{15}$的BASIC表达式。

四、表达式

所谓BASIC表达式，就是将符合BASIC程序规定的，用运算符和括号把数、变量、函数连接起来的数学式子。例如，2+SIN(3＊X)。在BASIC表达式中，所有符号都必须

一个一个并排地写在同一横线上，而不能在左上角上写次方或右下角上写下标，每一个字符都占用一格。

表2-1列举了一些代数式和BASIC表达式，以供比较。

表2-1

代数式	BASIC表达式	错误写法
2ab	2*A*B	2AB或2·A·B
a/-b	A/(-B)	A/-B
$(5x)^2$	(5*X)**2	(5X)**2
sin45°	SIN(45*Π/180)	SIN45
$\frac{a}{1+\frac{b}{2.7+c}}$	A/(1+B/(2.7+C))	A/(1+B/2.7+C)
$1+x+\frac{x^2}{2!}$	1+X+X**2/2	1+X+X**2/2!

第九节　键盘运算

在学习了BASIC语言中数的表示，标准函数、变量、运算符和运算规则以后，我们就能把代数式写成BASIC语言的表达式来进行运算了。如果属于复杂问题的运算，一般要编写程序后才能进行。由于BASIC系统具有对不带标号的语句，在按下ENTER键后就能立即解析执行的功能，因此对简单的算题，可以不通过编程序，而通过键盘打入不带标号的语句直接进行运算。这种由键盘直接进行的运算，称为键盘运算。由于它是通过键盘打入语句以命令形式进行的，所以又称为键盘语句命令。例如由键盘打入：

```
PRINT  5**2
```

按ENTER键后，在显示屏幕上立即显示出结果值25。

其实，键盘运算的主要用途是在调试程序时，用来检查中间结果和修改某些变量值，而不破坏原来的程序。

一、直接运算

通过键盘可以直接进行各种代数运算（如加、减、乘、除、乘方、开方、指数、对数等）、各种函数运算以及其它简单的数学计算。

【例1】 求5.3^2和$\sqrt{144}$的值。

利用键盘命令，打入PRINT 5.3＊＊2 按ENTER键后，显示出结果值28.09。

打入PRINT SQR144 按ENTER键后，显示出结果值12。

【例2】 求sin36°的值。

打入PRINT SIN（36＊π/180） 按ENTER键后显示出结果值0.56778525。

【例3】 计算$\sqrt{\dfrac{e^7}{8.5\times9.6}}$的值。

打入 A＝EXP 7 按ENTER键

打入 B＝8.5＊9.6 按ENTER键

打入 PRINT SQR（A/B） 按ENTER键后，显示出结果值3.6659422。

如果要知道根号内分子和分母的值，可以接着进行：

打入 PRINT A 按ENTER键后，显示1096.6332。

打入 PRINT B 按ENTER键后，显示81.6。

从这个例子可以看出，在键盘运算期间，计算机的内存贮器可以暂时保存变量值（如本例中的A和B），但不保存语句。

【例4】 计算$S=1^2+2^2+3^2+4^2+5^2$。

打入 PRINT 1+2*2+3*3+4*4+5*5按ENTER键后，显示出结果值55。

【例 5】 求解$x^2-8x+15=0$。

利用公式 $x_{1,2}=\dfrac{-B\pm\sqrt{B^2-4AC}}{2A}$

（当$B^2-4AC>0$时）

令 $Z=B^2-4AC$，

得 $x_{1,2}=\dfrac{-B\pm\sqrt{Z}}{2A}$

已知A＝1，B＝－8，C＝15。现用键盘运算求 X_1和 X_2的值，操作步骤如下：

（1）打入 Z=(-8)*(-8)-4*15 按 ENTER 键。

打入PRINT Z，按ENTER键后，检查 Z 值是否大于0。如果大于0便运算下去，否则按复根公式处理。

（2）打入PRINT (-(-8)+SQRZ)/2 按ENTER键后，显示出5，此值即为X_1的值。

再打入PRINT (8—SQRZ)/2，按ENTER后，显示出3，此值为X_2的值。

二、用在调试程序时检查中间结果和修改变量值

现仍以例3的内容来编写程序：

```
10  A=EXP 7
20  B=8.5*9.6
30  PRINT  SQR(A/B)
RUN
3.6659422
```

这个程序虽然简单，但可以说明如何用键盘运算去检查

中间结果和修改变量值。其方法如下：

检查中间结果A值和B值的操作过程是：打入PRINT A，B 按ENTER键后，显示出1096.6332和81.6，即A＝1096.6332，B＝81.6。

如果正在运行的是一个冗长、复杂的程序，并假设上列程序是这个复杂程序的一个程序段。要在调试程序时，检查A和B的值，则可以先加插一个程序行：25 STOP。再次使程序运行，当运行到25行时，由于STOP语句的作用，程序便自动暂停下来。此时就可以用上述方法，检查中间结果A和B的值。

如果想要试验一下改变A，B值会得到什么结果？可以用键盘打入新的A，B值。如：

打入 A＝EXP6 按ENTER键

再打入 B＝7.5＊8.6 按ENTER键

再打入 GOTO 30(或打入CONT) 按ENTER键，程序便以新的A，B值从30行继续往下执行。用这种方法，可以很方便地随时修改参数值，而不必修改程序。

值得注意的是，用键盘打入A，B的新值，并没有破坏或修改原来的程序，上例中10行和20行的内容仍然保存在内存中。这可以通过以下操作来证明。先打入RUN命令，使程序重新运行，当运行到25行时加入暂停语句，使程序暂停运行。这时检查A，B值，可以发现它们仍然是原来的值，而不是新值。也就是说，以键盘运算方式输入的无行号语句，在遇到RUN或NEW命令后便告失效。

通过上述例子可以看出：键盘运算不只是用来进行直接运算，而更主要的作用是在调试程序时，与键盘命令一起使用，从而成为检查程序，修改程序，排除程序故障的有效

工具。

习　题

〔2-1〕 将编好的程序上机操作时，分几个步骤？打入什么命令？

〔2-2〕 用键盘打入一个语句或命令后，如何把它们存入到内存器？

〔2-3〕 在R1微型计算机的键盘上，如何从一个键上得到四个键符？

〔2-4〕 在显示屏上，如何显示出从最小行号开始的BASIC程序和从n行号开始的BASIC程序？

〔2-5〕 怎样从一个BASIC程序中，消去一个语句？

〔2-6〕 用指数记数法表示下列各数：

（1）8657342190

（2）-0.000000123

〔2-7〕 指出下列BASIC表达式的错误，并改正之。

数学式子	BASIC表达式
（1）x^{y-1}	X**Y-1
（2）$x^{2x}y$	X**2*X*Y
（3）cos64°	COS64
（4）$a+bc$	A+BC

〔2-8〕 试用键盘直接输入数据，求下列各式之值

（1）$5+3\sqrt{2.4^2+1}$

（2）$\dfrac{(286-64)\times 2^6}{1.2\times e^4}$

第三章　赋值语句和输入输出语句

第一节　打 印 语 句

在BASIC语言中，几乎每个源程序都要用到打印语句(PRINT语句)。PRINT语句可以命令计算机将计算结果或数据显示在屏幕上。怎样操作呢？当然可以逐个地按P、R、I、N、T这五个键，使在显示屏的左下角上出现PRINT字样。但一般计算机都设有专用的PRINT键，R1微型机的专用PRINT键是按SHIFT键和Z键来实现的，而有些机则用"？"键来代替PRINT键。

一、PRINT语句的一般格式和用途

1．一般格式

PRINT‹一组打印项›

语句中打印项可以是数、变量、函数或表达式等。

2．功能

（1）在显示屏幕上显示出各个打印项（常数、变量、函数或表达式）的值，而不是打印出变量、函数或表达式本身的字样；同时变量要事先被赋值，否则在屏幕的左下角将显示出错报告信息UV　IN‹行号›，指出该行号语句中有未定义变量的错误。

例如，用打印语句打印如下程序：

```
10   PRINT   5
```

```
20  PRINT  - 5
30  X=3.1416
40  PRINT  2 * X
50  PRINT  COSX
60  PRINT  Y
RUN
```

则在显示屏幕上显示出：

```
5
- 5
6.2832
- 1
```

程序运行到10行时，打印出 5，这里正的符号不打印，正符号位也不留空格。程序运行到20行时，打印出 - 5，负号占了一格位置。运行到40行时，打印出表达式的值。运行到50行，同样是打印出余弦函数值。由此可见，PRINT 语句具有计算和输出双重功能，在遇到表达式时，它先进行计算然后打印输出。运行到60行时，由于没有给变量Y赋值，此时在显示屏下方出现UV IN 60的报告信息，指出60行语句中有未定义变量的错误。

（2）如果要求把程序中的字符串按原样照印出来，则要用引号把这些字符串括起来。这样，计算机在运行时，就会将引号内所有字符原原本本地照印出来，但不显示引号。例如：

```
10  X= 5
20  PRINT  "ABCD"
30  PRINT  "X=";  X
RUN
```

ABCD

X = 5

程序执行结果，打印语句把引号内ABCD和X=原样照印出来了。

二、打印输出格式及其标点符号的规定

在打印一批数据时，为使打印结果清晰起见，要求有不同的打印输出格式。

打印输出格式有两种：分段格式（又称标准格式）和紧凑格式。现分别讨论如下。

1．分段格式

采用分段格式时，语句中各打印项之间的标点符号规定用逗号“，”来标志。

分段格式是将一个显示行划分为若干段，每段第1格称为标准位置，或者说作为标准定位，每个字符占一格位置，而每段占有多少格呢？不同的计算机有不同的规定。例如：

R1微型计算机的一个显示行为32格，分为二段，每段占16格。

APPLE Ⅱ微型机分为三段，第一、二段占16格，第三段占8格。

TRS-80微型机分为四段，每段占16格。

Commodere微型机分为十段，每段占8格。

现以R1微型机为例，分析下面程序的打印结果。

```
10  PRINT  2, "ABC"
20  PRINT  "ABC", 2
30  PRINT  1, 2, -3, -4, 5,
40  PRINT  6
RUN
```

```
2          ABC
↑          ↑
ABC        2
↑          ↑
1          2
↑          ↑
-3         -4
↑          ↑
5          6
↑          ↑
```

从上列程序的运行结果可见：

（1）符号↑表示每段的起始标准位置，在屏幕上不显示出来。

（2）逗点的作用。在执行打印语句时，遇到逗号，计算机就自动地将下一个打印项移至下一打印段的标准位置后才开始打印。例如，30行的-3和5都被移到下一打印段的标准位置。

（3）如果一行中的各段均已打满（如30行打印语句），而还要继续按分段格式打印时，则自动换行转至下一行第一段接着打印。

（4）如果句末是逗号，则仍然表示要按分段格式打印。因此执行到下一个打印语句时，应在原先打印行上按分段格式接着打印下去。例如30行句末有逗号，这时40行中的6不是从显示行的第一段开始，而是接在30行的第二段打印出来。

（5）如果句末无任何标点符号，则执行下一个打印语句时将自动换行，即从第一段开始打印。

2．紧凑格式

采用紧凑格式时，规定各打印项之间用分号“；”来隔开。打印时遇到分号，它表示打印项之间的衔接紧凑，即后

一项数据（或字符串）紧接前一个打印项打印出来。至于项与项之间是否留空格，不同的计算机有不同的规定。R1微型机和APPLE Ⅱ微型机规定不留空格。TRS-80微型机规定在打印数据时，数的前面留一格作为符号位，但只打印负号，而不打印正号；数的后面规定留一空格；如果打印两个字符串则其间不留空格。

现将下面程序中的打印语句在R1微型机和TRS-80微型机上的打印结果分别表示出来，以资比较。

```
10  PRINT  1; 2; -3; -4
20  PRINT  "26+14="; 26+14;
30  PRINT  3; 4
40  PRINT  "GOOD"; "MORNING"
50  PRINT  "GOOD"; "␣"; "MORNING"
60  PRINT  "GOOD  MORNING"
```

在R1微型机上运行的结果如下：

```
12- 3 - 4
26+14=4034
GOODMORNING
GOOD  MORNING
GOOD  MORNING
```

在TRS-80微型机上运行的结果如下：

```
 1    2   -3   -4
26+14=  40    3    4
GOODMORNING
GOOD  MORNING
GOOD  MORNING
```

当打印语句末尾是分号时，仍然按紧凑格式打印，即运

行到下一个打印语句时，应在原先打印行上按紧凑格式接着打印。所以上述程序的20行和30行打印语句，在R1微型机上运行的结果是26＋14＝4034。如果不注意打印输出格式，此处就容易被认为是26＋14等于4034。为了使这些数值之间有空格隔开，应在20行语句的末尾和在30行语句3与4之间，插入空格字符打印项。这样，其打印结果为：26＋14＝40　3　4。

三、无语句体的打印语句

在PRINT的后面没有打印项（数据、符号或字符串）的打印语句，称为无语句体的打印语句。它的作用可以分两种情况来讨论。

1．如果前一个打印语句的句末无标点符号，则此无语句体的打印语句起了空行的作用。例如：

```
10  PRINT  "A"; "B"
20  PRINT      （起空一行的作用）
30  PRINT  1; 2
RUN
AB


 1   2
```

由于10行语句的末尾没有标点符号，故20行无语句体的打印语句起了空一行的作用。

2．如果前一个打印语句句末标有逗号或分号时，则这个无语句体的打印语句起了换行作用。例如：

```
10  PRINT  "A"; "B";
20  PRINT      （起换行作用）
30  PRINT   1; 2
RUN
```

```
A B
 1 2
```

由于10行语句句末是分号，故20行无语句体打印语句起换行作用。

在本程序中，如果没有20行打印语句，则A B12在同一行按紧凑格式打印出来。

如果在一个程序中，连续出现两个以上无语句体的打印语句，情况又怎样呢？请看看下列程序。

```
10  PRINT  "A"; "B";
20  PRINT       （起换行作用）
30  PRINT       （起空一行作用）
40  PRINT   1;  2
RUN
A B

1  2
```

从本程序执行结果可见，20行语句起换行作用，30行语句起空一行的作用。同理，每多一个无语句体的打印语句，就多空一行。

四、打印语句不能兼作赋值语句

例如，要求把5赋给变量X并打印出X值来。如果写成下面形式：

```
10  PRINT   X=5
```

则这种写法是错误的。因为打印语句不能兼作赋值语句。正确的写法应该是

```
10   X = 5
20  PRINT   X
```

```
30 PRINT "X="; X
```

还有一些特殊的打印输出格式，将在有关章节中进行讨论。

第二节 赋值语句

赋给变量一个具体数值的语句，叫做赋值语句。

一、为什么要对变量赋值

电子计算机只能对数值进行计算，而不能直接进行变量运算。例如，计算机无法对X * X和X * X－Y * Y进行运算，它不能象解代数题那样把X * X写成X^2，把X * X－Y * Y分解为(X＋Y) * (X－Y)。而必须先给变量一个具体数值，计算机才能进行运算。

赋值语句的格式为

LET(〈变量〉)＝〈表达式〉

这里LET是语句定义符，变量可以是简单变量、下标变量或字符串变量。表达式可以是算术表达式，也可以是字符串表达式。（关于下标变量和字符串变量的内容详见第七、八章。）例如：

```
10 LET X=2
20 LET Y=1
30 LET A=X*X
40 LET B=Y*Y
50 LET C=A-B
60 PRINT A, B, C
RUN
```

4　　1
3

上面程序的10行和20行是赋值语句，它们运行的结果是将数值2赋给变量X，将数值1赋值给变量Y，从而使30行中的X * X、40行中的Y * Y变为具体数值的运算，然后再将运算结果值分别赋给变量A和B。同样，50行赋值语句也是先进行数值运算（A－B），然后赋值给变量C的。最后再通过打印语句输出A，B和C值。

二、赋值语句的含意

1．赋值语句是用来对一个变量赋给一个确定的数值的语句。应该注意，一个语句只能对一个变量赋值，而不能同时对两个以上的变量赋值。例如，把数值2赋给X变量，数值1赋值给Y变量，不能写成LET　X，Y＝2，1或LET　X＝2，Y＝1。

实际上，赋值语句含有“传送”的意思。即用语句中等号左边的变量代表计算机中的一个内存单元，而把等号右边的数值（或运算后的数值）传送到变量的内存单元中去。这样，上面程序的含义是

2⇒X，1⇒Y，X * X⇒A，Y * Y⇒B，A－B⇒C。式中⇒表示传送的意思。

2．赋值语句中的等号“＝”叫赋值号，它与数学上的等号意义不同。在数学上，X＝2表示等式两边完全相等，也可以写成2＝X。而在BASIC语言中，10　LET　X＝2，其含意是将2送给X，不能写成10　LET　2＝X，因为这样是把X赋给2，那就完全没有意义了。

请看下面两个程序（注意两个30行语句的区别）。

程序（1）　　　　程序（2）

```
10  LET   X= 3          10  LET   X= 3
20  LET   Y= 4          20  LET   Y= 4
30  LET   X= Y          30  LET   Y=X
40  PRINT X,Y           40  PRINT  X,Y
RUN                     RUN
 4        4              3        3
```

在数学上，X=Y完全可以写成Y=X。但在程序（1）中，30句的含意是将Y的值（4）赋给X变量，所以X变量重新赋值为4，而Y值不变，故打印结果为4　　4。在程序（2）中，30行语句体写成Y=X，是将X的值（3）重新赋值给Y，而X值不变，故打印结果为3　　3。

又如在BASIC 程序中，常常可以看到下面这种形式：

LET　X=X+1

我们知道，在代数中这个式子是不成立的，但在BASIC程序中，它表示先进行等式右边的运算(将X原有的值加1)，然后将结果送回X中。

3．赋值语句具有先运算后赋值的功能，因此等号右边可以是变量、函数或表达式。只要表达式中的变量在语句出现之前被赋值，就可以进行数值运算，并将运算结果再赋给等号左边的变量。所以在BASIC程序中，简单的运算都采用赋值语句。

三、赋值语句中的一些规定

明白了赋值语句的含意后，不难理解下述的一些规定。

1．若多次给一个变量赋值，则变量被赋为最后一次的值（或称为以新换旧）。例如：

```
10  LET   A= 1
20  LET   A= 2
```

```
30  LET  A = 3
40  PRINT  A
RUN
 3
```

2．变量之间可以辗转赋值。例如：

```
10  LET  A = 3
20  LET  B = A
30  LET  C = B
40  PRINT  A, B, C
RUN
 3      3
 3
```

上述程序把3⇒A⇒B⇒C，相当于3⇒A，3⇒B，3⇒C。

3．在基本BASIC程序中，一个赋值语句只能给一个变量赋值。但在扩展BASIC程序中，一行可写几个语句，语句间用冒号分隔。例如：

```
10  LET  A = 3
20  LET  B = 4
30  LET  C = 5
```

若写成10　LET　A = 3，B = 2，C = 5，则是错误的；但在扩展BASIC程序中，可以写成10　LET　A = 3 ：LET B = 4 ：LET　C = 5

有些微型计算机容许省略LET。这时，上述第10行语句就可以写成

```
10  A = 3 : B = 4 : C = 5
```

R 1 微型机规定一个程序行只能写一个语句，但LET可以省

略不写。

4. 赋值号的左边必需是一个变量，而不能有多个变量。下面的写法是错误的：

```
10 LET A+B=2+3
```

5. 赋值语句中不能连用几个赋值号。下面的用法是错误的：

```
10 LET Z=Y=X=3
```

6. 在编程时，赋值语句等号的右边应尽量避免重复计算。例如计算下式的值：

$$y=7\sin^3x-6\sin^2x+5\sin x$$

其程序可写成

```
10 LET Y=7*SINX**3-6*SINX**2+
5*SINX
```

虽然上式的运算结果是正确的，但每项都要计算SINX的值，速度就慢了。

如果将10行语句改写成

```
10 LET A=SINX
20 LET Y=7*A**3-6*A**2+5*A
```

则只需进行一次SINX的运算，就可以得出结果。如果把20行语句进一步改写成

```
20 LET Y=(7*A**2-6*A+5)*A
```

则乘法运算的次数就更少了。显然，执行这个程序所需要的时间便可以进一步缩短。

最后说明一下，某些计算机对未赋值的变量作零处理，而R 1 微型机在执行程序中，遇到未赋值的变量时则报告出错信息。

请看下面的程序：

```
10  LET  A= 3
20  LET  B= 4
30  LET  C= 5
40  LET  X=A+B+C
50  PRINT  X
RUN
12
```

执行完50行语句后，变量X的值为12，但如果只执行20行语句，例如：

RUN 20

则在显示屏幕的左下角将出现出错报告信息UV IN 40，其意思是在40行语句中出现了未定义的变量，即X变量未被赋值。

第三节 键盘输入语句

用LET语句给变量赋值，虽然方法简单，但有时并不合用。例如需要编写一个商店计价的程序，由于顾客购买货物的数量和单价都不是固定值，因此使用赋值语句就很难给变量赋值了，但如果改用键盘输入语句就非常适合。

键盘输入语句不是在编程时给变量赋值，而是在程序运行之后，由使用者从键盘打入数据给变量赋值的。它为用户提供了一种通过人机对话输入变量值的方法，从而增加了程序设计的灵活性。

键盘输入语句的格式是

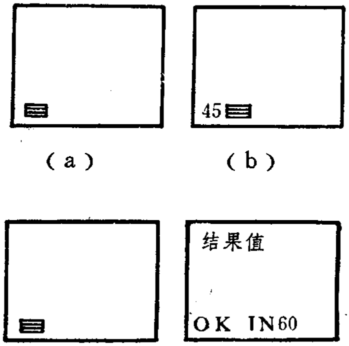
INPUT ‹变量名›

例如，商店售货计价程序如下：

```
10  T=0
20  INPUT  A      （用键盘输入售货数量A）
30  INPUT  B      （用键盘输入货物单价B）
40  C=A*B         （价钱=数量×单价）
50  T=T+C         （累加金额）
60  PRINT  T
```

当程序运行到20行INPUT语句时，便暂停执行（动态停机），并在显示屏幕的左下角出现闪动光标，如图3-1（a）所示，它表示等待输入数据。此时，如果通过键盘打入数据45，则在屏幕的左下角显示出45，而且闪动光标相应右移二格至数据的右边，如图3-1(b)所示。在按ENTER键后，就将数据45赋给变量A了，同时计算机继续执行程序。由于下一个语句（30行语句）又是INPUT语句，则闪动光标重新返至屏幕左下角的起始位置，如图3-1（c）所示，它仍然表示等待输入数据。这时，再通过键盘打入一个数据，接着按ENTER键，则此数据便赋给变量B，同时继续执行后面的程序。由于以后再无遇到INPUT语句，故不再出现动态停机，等待输入数据的情况，直至执行到60行打印语句时，将结果值打印在显示屏幕上，如图3-1（d）所示。

执行INPUT语句时，显示屏上出现画面的情况，随计算机的不同而有所不同。有的计算机显示出一个“？”，意思是向使用者询问INPUT语句中变量的值是多少？使用者通过键盘输入数据后，按回车键（或ENTER键），程序又继续执行，如果又遇到INPUT语句时，再显示出一个“？”……因此，上列程序运行的过程如下：



（a）（b）（c）（d）

图3-1 执行INPUT语句时显示屏幕上的画面

RUN ENTER

? 45 ENTER ? 2 ENTER 90 （结果值）

【例1】编写一段求并联电阻电路的电流值的程序。计算公式为

$$\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4}$$

$$I=U/R=U\left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4}\right)$$

程序编写如下：

```
10  INPUT  R1
20  INPUT  R2
30  INPUT  R3
40  INPUT  R4
50  INPUT  U
60  I=U*(1/R1+1/R2+1/R3+1/R4)
```

```
70  PRINT  I
RUN  ENTER
```

设$R1$为100Ω，$R2$为150Ω，$R3$为240Ω，$R4$为360Ω，U为24V。

打入100　按ENTER

打入150　按ENTER

打入240　按ENTER

打入360　按ENTER

打入24　按ENTER

最后屏幕显示出0.56666667。

上述程序共有五个变量，用了五个INPUT语句。当打入RUN后，显示屏的左下角出现第一个闪动光标，表示等待R1值的输入；打入100后按下ENTER键，又出现第二个光标，等待R2值的输入，……直至全部输入完毕后，计算机才进行运算，输出最后结果并显示在屏幕的左上角位置。

若想要改变上述五个变量中的一个或多个数据时，可以不修改程序。而使用INPUT语句，使原程序再运行一次，并从键盘输入另一组数据，这样，就可以得到新的结果。如果要计算几组数值，令程序运行几次即可。

由此可见，使用INPUT语句，对于输入多组变量是比较灵活方便的。对于调试程序或试验程序，往往需要多次改变参数，以便选择更优方案，这时使用INPUT语句则是最合适的了。

使用INPUT语句时，从键盘输入的数只能是一个具体的数值，它可以是整数、小数，但不能是分数、函数或表达式。

如果程序中输入语句的变量太多，为了便于辨别打入的

数据是送给那一个变量的，可以用PRINT语句加以说明。这时，例 1 的程序改为

```
5   PRINT "R1␣␣␣";"R2␣␣␣";"R3␣␣␣";
"R4␣␣␣"; "U"
10  INPUT  R1
15  PRINT  R1; "␣␣";
20  INPUT  R2
25  PRINT  R2; "␣␣";
30  INPUT  R3
35  PRINT  R3; "␣␣";
40  INPUT  R4
45  PRINT  R4; "␣␣";
50  INPUT  U
55  PRINT  U
60  I = U * (1/R1 + 1/R2 + 1/R3 + 1/R4)
65  PRINT
70  PRINT  "I = "; I
RUN
R1        R2        R3        R4        U
100        150        240        360        24
I = 0.56666667
```

最后需要说明的是，对R1微型机来说，一个 INPUT语句只能输入一个变量。而有些计算机，则允许在一个INPUT语句中输入若干个变量，但每个变量之间要用逗号“，”分隔开来，而句末不能加逗号。

执行 INPUT 语句时，用键盘输入数据的方法有两种：一种是一次只输入一个数据，然后按回车键，直至输完所有

数据为止；另一种是输入完所有的数据后，才按回车键，但每个数据之间要用逗号分隔开来。详细的操作方法，请参阅所使用的计算机使用说明书，这里就不赘述了。

第四节　读数语句、置数语句和恢复数据区语句

在讨论本节内容以前，有必要说明一下，R1微型机的BASIC程序是没有本节所介绍的语句的，但鉴于如下三方面原因，我们仍以一节篇幅来介绍它。

（1）读数语句/置数语句（READ/DATA语句）是BASIC语言中一个很基本的语句。它在程序中经常出现，特别是在处理大批数据时，用了它可使程序简化和应用灵活。

（2）很多微型计算机的BASIC系统都具有READ/DATA语句，学习了这些语句，就更便于使用其它类型的计算机了。

（3）既然READ/DATA语句如此重要，而R1微型机又不具备这些语句，在编程时往往感到很不方便，但只要真正掌握了READ/DATA语句的使用方法，就可以用INPUT语句，再配合下标变量和循环语句、GOTO命令等，来代替READ/DATA语句。这样既掌握了READ/DATA语句，又加深了其它各种语句的使用知识的了解。

一、READ/DATA语句的作用

在向计算机输入大批已知数据时，如果再用INPUT语句就太麻烦了。因为每执行一次INPUT语句，计算机都要暂停程序一次，等待输入新的数据，第二次执行程序时，又要重新输入数据。因此提出了给变量赋值的第三种方式，即读数和置数（READ/DATA）语句。

READ/DATA语句的格式是

READ ‹变量组›

DATA ‹数据组›

在变量组和数据组内，用逗号“，”分隔开每一个变量和数据值，末尾不加逗号。READ/DATA 语句的功能，是从DATA语句的数据组中，逐个地把数据读入到READ语句中对应位置的变量中。

现在我们仍然以上节讨论过的求并联电阻电路的电流值为例，用 READ/DATA 语句改编一段程序：

```
10 READ R1, R2, R3, R4, U
20 DATA 100, 150, 240, 360, 24
30 DATA 50, 110, 200, 30, 20
40 LET I=U*(1/R1+1/R2+1/R3+1/R4)
50 PRINT "I="; I
60 GOTO 10
RUN
I= .566667
I=1.34848
```

当输入程序时，DATA语句中的一组数据，就已经由BASIC解析程序将它们依次地送入到计算机的内存器的数据区中了，即在程序正式运行前，DATA语句的全部数据已经写入到数据区了。所以DATA 语句是一种非执行语句，它不会使计算机产生任何操作，它的任务只是为READ语句提供数据。

既然 DATA 语句是在输入程序时就已经将数据存入内

存器了，因此一组数据无论是用一个 DATA 语句或用几个 DATA 语句输入，也无论 DATA 语句是放在程序的开头还是末尾，只要数据的顺序不变，产生的效果是一样的。所以，DATA语句可以放在程序中的任何位置，也可以把它分为几个 DATA 语句。通常为了使程序清楚起见，一般将它放在程序的开头或者末尾，但放在末尾更便于修改数据。

READ语句是一种可执行语句。当程序运行到READ语句时，把DATA语句中（或数据区中）第一个数据赋给READ语句的第一个变量，第二个数据赋给第二个变量，即依次将数据赋给READ语句中对应的每个变量。例如上列程序中，将第一个数据100赋给R1，第二个数据150赋给R2，240赋给R3，360赋给R4，24赋给U。运行到GOTO语句时，程序就无条件地转去执行GOTO语句中所指的行号为100的语句，即又执行READ语句。它继续往下读出数据，将50赋给R1，110赋给R2等等。再次执行GOTO语句转到READ语句时，由于此时无数据可读，在计算机屏幕的左下角显示出错信息。

由上所述，我们不妨设想，在DATA语句中的下面有一个所谓"数据指针"（实际上是不存在的），它如下所示：

100，150，240，360，24
↓

指针的初始位置指向第一个数据的前面。当READ语句中的第一个变量需要读数据时，就读指针后面的第一个数据100。当读完该数据后，指针就向右移动到第二个数据的前面。即

100，150，240，360，24
　　　↓

第二个变量需要读数时，就读指针后面的数据，即第二个数

据150，依此类推。

二、关于READ/DATA语句的几项规定

1. DATA语句与READ语句必需成对出现。而一个DATA语句或一个READ语句，则可以分成两个或多个DATA语句及READ语句。例如：

程序（1）

```
10  READ  X, Y, Z, A, B
20  DATA  1, 2, 3, 4, 5
30  LET  D=(X+Y+Z)/(A+B)
```

程序（2）

```
10  READ  X, Y
20  READ  Z, A
30  READ  B
40  LET  U=(X+Y+Z)/(A+B)
50  DATA  1, 2, 3
60  DATA  4, 5
```

程序（3）

```
10  DATA  1
20  READ  X, Y, Z, A
30  DATA  2, 3
40  READ  B
50  DATA  4, 5
60  LET  U=(X+Y+Z)/(A+B)
```

程序（2）把程序（1）中的READ/DATA语句写成三个READ语句和两个DATA语句，而程序（3）则写成两个READ语句和三个DATA语句，且DATA语句的位置不受限制地随便放置。这三个程序是完全等效的，因为它们

当中变量和数据的顺序和个数并没有改变。

2. DATA语句中数据的个数必须大于或等于READ语句中变量的个数。否则作为出错处理。

3. DATA语句中各个数据之间和READ语句中各个变量之间都要用逗号“，”分开，但一行的最后不能加逗号。

4. DATA语句中的数据只能是具体的数值，不能是分数、函数或表达式。

三、恢复数据区语句

在含有READ/DATA语句的程序中，经常伴随出现恢复数据区语句。

它的格式是

RESTORE

语句定义符RESTORE的后面为空白。这个语句的作用是重复使用数据区中的数据，或者是将“数据指针”恢复到起始位置，使执行到下一个READ语句时，可以从第一个数据读起。例如：

```
10  READ  A, B, C, D, E
15  RESTORE
20  READ  M, N, O, P, Q
 ⋮
100  DATA  1, 2, 3, 4, 5
```

如果上述程序中无RESTORE语句，则执行到20行READ语句时将无数可读。现在加插了一句恢复数据区语句15 RESTORE，情况就不同了。它使数据指针恢复到起始位置，即指在DATA语句中第一个数据“1”的前面，从而使执

行20行READ语句时，又依次读出DATA语句中五个数据1，2，3，4，5，并赋给20行READ语句中五个变量M，N，O，P，Q。例如：

```
10  READ  X
20  RESTORE
30  READ  Y, Z
40  PRINT  X, Y, Z
50  DATA  50, 60
RUN
50    50
60
```

如果只希望重复使用数据区中某一部分数据，而不需要使用全部数据时，则可以采用假读的办法，即在READ语句中增加一些称为虚变量的无用变量。虚变量在READ语句中同样也会被赋值。这样，便起到跳过了不需要重复使用的数据的作用。例如：

```
10  READ  A, B, C, D, E
15  RESTORE
20  READ  T, T, F, G, H  （T为虚变量）
25  RESTORE
30  READ  T, W, X, Y, Z  （T为虚变量）
40  PRINT  A, B, C, D, E, F, G, H, W, X,
Y, Z
50  DATA  1, 2, 3, 4, 5
RUN
1  2  3  4  5  3  4  5  2  3  4  5
↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑
```

A　B　C　D　E　F　G　H　W　X　Y　Z

四、三种输入数据语句的比较

LET 语句的主要特点是：一个语句只能给一个变量赋值；可以用变量和表达式给变量赋值；同时具有赋值和运算的功能。它适用于变量少和不需要重复赋值的程序。它是一种进行算术运算的主要语句。

READ/DATA 语句的主要特点是：一个语句可赋给多个变量，借助于循环语句或GOTO语句，可以使多个数值赋给多个变量。它适用于成批已确定的数据的输入。

INPUT 语句的赋值方式，是通过键盘输入具体数值。它适用于给变量输入不固定的数值以及参数变化，或者需要进行边运算、边修改、边比较的程序中的输入语句。

习　题

〔3-1〕　指出下列BASIC语句的错误之处，并改正之。

```
50  X + 1 = Y
20  INPUT  A,
30  INPUT  B,
10  READ  A; B
80  PRINT  X = Y + 2
60  "X + Y = "; X + Y
```

〔3-2〕　下面每一个BASIC程序里都有一个错误，试指出其错误之处。

```
(1) 10  READ  A, B
    20  PRINT  A + B
```

```
     99  END
(2)  10  LET  P= 5
     20  LET  Q=22
     30  LET  S=(P+Q)/(Q-R)
     40  PRINT  M, N, S
     99  END
```

〔3-3〕 写出下面程序的运行结果，并指出后面三个赋值语句的作用。

```
10  A = 1
20  B = 2
30  PRINT  A, B
40  C = A
50  A = B
60  B = C
70  PRINT  A, B
```

〔3-4〕 用分段格式与紧凑格式打印下列数据。

(1) 1，3，5，7，2，4，6，8

(2) -1，2，-3，4，-5，6

〔3-5〕 计算 $I=2$ 安，$R=25$欧时的U值，输出格式$U=\cdots$V。

〔3-6〕 编一个计算 $f=\dfrac{1}{2\pi\sqrt{LC}}$的程序。

当$L=0.012$亨，$C=2.2\mu$F

$L=0.002$亨，$C=1.0\mu$F

$L=0.0004$亨，$C=0.002\mu$F

时的f值。

〔3-7〕 用INPUT语句写出华氏温度转换成摄氏温

度的程序。计算公式为

$$C=(F-32)/1.8。$$

〔3-8〕 用INPUT语句编写一程序，求四个学生四门功课的每人平均成绩。

学生＼科目	A	B	C	D	V（平均）
甲	××	××	××	××	××
乙	××	××	××	××	××
丙	××	××	××	××	××
丁	××	××	××	××	××

〔8-9〕 编写如下一个问题的程序：已知鸡兔同笼时总头数为H，总脚数为F，求鸡兔各多少？

第四章　无条件转向语句和条件转向语句

BASIC程序的运行，一般是按行号由小到大的顺序执行的。但是，如果需要的话，也可以通过在程序中安排转向语句来改变这种执行顺序，从而转移执行所指定的语句。

转向语句有两种：一种是无条件转向语句（GOTO语句），另一种是条件转向语句（IF-THEN语句）。

第一节　无条件转向语句

GOTO语句的作用是，当程序执行到此语句时，无条件地转移到语句中所指定的程序行去继续执行。例如，要求每输入一个数，接着把它打印出来，就可以用GOTO语句编出下列程序：

```
10  INPUT  X
20  PRINT  X
30  GOTO  10
```

程序中10行和20行的内容，是从键盘输入一个数再把这个数打印出来。当程序执行到30行时，由于遇到GOTO语句，因而无条件地转回到10行，又继续执行INPUT语句和PRINT语句，如此循环下去。

GOTO语句的格式为

GOTO〈行号〉

GOTO前面的行号是程序行的标号，而其后面的行号则是要去执行的那个程序行的行号，因此GOTO后面的行号必需是程序上存在的行号，否则，程序执行到GOTO语句时就会“迷路”。这个行号，可以在GOTO语句之前，也可以在GOTO语句之后。放在GOTO语句之前，意思是使程序转回到前面所指示的语句继续执行；放在GOTO语句之后，则是跳过一段程序而转到后面所指定的行号去执行。由此可以看出，使用GOTO语句可以方便地改变程序的执行顺序。

例如，有一段累加运算的程序：

```
10  T = 0
20  INPUT  X
30  PRINT  "X = "; X,
40  T = T + X
50  PRINT  "T = "; T
60  GOTO  20
RUN
X = 1       T = 1
X = 2       T = 3
X = 3       T = 6
```

程序中40行的内容是累加算式T = T + X。算式等号左边的T是本次的累加总和，算式等号右边的T是前一次的累加总和。T的起始值必须是零，应写在程序的开始，如果第一次输入的数据是1，则T = 1。执行到60行时，程序无条件地转回20行去继续执行。设第二次输入数据是2，则累加上去，T的新值为3，…，如此反复执行下去，便可以得到所需要的累加数。

如果希望程序停止执行，可以按下特殊功能键BREAK，

使程序中断在20行。也可以置一个“输入终止标志”（本例假设为零），当键盘输入零值时表示输入结束，程序立即停止执行。所以上例的程序可以改写成如下程序：

```
10  T = 0
20  INPUT  X
30  PRINT  "X=", X,
35  IF  X=0  THEN  GOTO  70
40  T=T+X
50  PRINT  "T="; T
60  GOTO  20
70  STOP
```

怎样实现程序暂停呢？我们知道，在本程序中60行语句是无条件转向语句，当程序执行到60行时，就无条件地转到前面20行去继续运行，而35行是条件转移语句，当满足条件X = 0时，就转去执行后面70行语句。由于70行是暂停语句，因此就使程序暂停执行。

第二节　条件转向语句

一、条件转向语句的作用

条件转向语句（IF–THEN语句），简称为条件语句。它的作用是：当程序运行到此语句时，如满足语句中给定的条件就转移，如果条件不满足则仍然按顺序运行。

例如，从输入一组数据中，把正数挑选出来并打印输出。程序如下：

```
10  INPUT  X
20  IF  X<=0  THEN  GOTO  10
```

```
30  PRINT  X
40  GOTO  10
```

本程序中20行的内容是：如果输入的X值小于或等于零，则转回去执行10行语句。这里X＜=0就是给定条件，如果不满足给定条件（即X是正数），则程序按顺序运行，并打印出正数。接着运行到40行，这时又无条件地返回去执行10行语句。如此反复循环，就把输入的正数都打印出来了。

上述程序还可以简写成下面三个语句，而其功能不变：

```
10  INPUT  X
20  IF  X>0  THEN  PRINT  X
30  GOTO  10
```

由此可见，条件语句是一个很重要的语句，它可以使计算机根据程序所给定的条件，对运行中的变量值进行判断，这个判断包括数值上的比较判断和逻辑上的判断，然后由判断结果作出不同的处理。正因为有了这个条件转移语句，才使得计算机具有判断能力。

二、R1微型机的条件语句格式

R1微型机条件语句的格式是

IF‹关系式› THEN‹语句›

具体又分为两类：

行号　IF‹关系式›　THEN　GOTO　行号；

行号　IF‹关系式›　THEN　语句。

因此，条件语句主要包括语句定义符（IF-THEN）、关系式和转向语句。

1．语句定义符IF-THEN

IF-THEN的含义是：如果……则……；或者是：如果

……就……。

2．关系式

关系式的一种形式是：

（变量，表达式）〔比较符〕（数，变量，表达式）

其中，比较符号有以下几种：

= 等于

< 小于

> 大于

<> 不等于

>= 不小于

<= 不大于

关系式又称判别式，它用来指出给定的条件。例如：

10 IF A>=3 THEN GOTO 60 （变量与数值比较）

10 IF X<Y THEN GOTO 30 （变量与变量比较）

10 IF T<=A**3 THEN GOTO 70 （变量与表达式比较）

10 IF 3*A*X<>5*LOGA THEN GOTO 50 （两个表达式比较）

如果给定的条件是“<>0”的话，则比较符号可以省略不写。例如：

10 IF X**Y<>0 THEN GOTO 50

可简化为：

10 IF X**Y THEN GOTO 50

又如：10 IF X<>0 THEN GOTO 50

可简化为：

10 IF X THEN GOTO 50

关系式的另一种形式是逻辑判断式：

（比较式）[逻辑运算符]（比较式）

例如：

10 IF X>0 AND X<6 THEN GOTO 100

20 IF X=6 OR X=10 THEN GOTO 200

30 IF X>6 AND X<10 THEN GOTO 300

其中 AND 为逻辑“与”的运算符号，OR为逻辑“或”的运算符号。10行语句的含意是：如果X>0和X<6这两个条件同时成立（即X=1～5），就转去执行100行语句。20行语句的含意是：X=6或者X=10这两个式子中只要有一个条件成立，就转去执行200行语句。上列三行的内容实际上是：

当X=1～5时，转去执行100行语句；

X=6或X=10时，转去执行200行语句；

X=7～9时，转去执行300行语句。

在条件语句中，跟在 THEN 的后面的语句也可以是条件语句。例如：

40 IF X>0 THEN IF X<6 THEN GOTO 100 其含意与10行内容完全相同。

3．转向语句

跟在 THEN 后面的语句，称为转向语句。它指出计算机应该转向去执行什么语句。除了DEF、REM、DATA、CONT、FOR—NEXT语句以外，所有BASIC 语句都可以

接在THEN的后面作为转向语句使用。

应该指出，在R1微型机的条件语句中，THEN后面一定要跟一个语句，而且只能跟一个语句，这个语句也包括条件转移语句。例如，很多计算机都允许把程序写成

```
10 IF X>0 THEN 100
```

这个语句的含意是：如果X>0的条件成立，就转去执行100行语句。但R1微型机规定THEN的后面应该是一个语句，故上例应改写为

```
10 IF X>0 THEN GOTO 100
```

不过，不要误认为在R1机的条件语句中，THEN的后面一定要加GOTO才行。如：

```
10 IF X>0 THEN PRINT X
```

这个语句的含意是：如果X>0的条件满足，就转去打印X值。这里PRINT X已经是一个语句，如果THEN后面还加上GOTO，那就错了。如：

```
10 IF X>0 THEN GOTO PRINT X
```

这种写法是不允许的。

三、IF-THEN语句的各种格式

IF-THEN语句的格式较多，常见的有下列四种类型。至于哪一种格式适合你所使用的计算机，那就要看机器的BASIC解析系统来作选择了。使用之前，应该参阅所用计算机的使用手册，并通过上机操作进行试验。

1．第一种类型

IF……THEN 〈行号〉

或 IF……GOTO 〈行号〉

前面已经指出过，R1微型机不能直接采用这种格式，应改写为

行号　IF……THEN　GOTO　‹行号›

2. 第二种类型

行号　IF……THEN　‹语句›

R1微型机可以直接采用这种形式，它是R1机唯一能够采用的条件语句格式。

3. 第三种类型

行号　IF……THEN　‹行号或语句›　ELSE‹行号或语句›

其含意是：如果条件满足，就转去执行THEN后面跟着的行号或语句；如果条件不满足，就转去执行ELSE后面跟着的行号或语句。例如：

```
IF  X>0  THEN  PRINT  X  ESLE  X=ABS
(X)
```

此句的意思是：如果X是正数就转去打印出X的值，否则，转去执行取绝对值的赋值语句，接着执行条件语句的下一个语句。

注意：R1微型机不能使用这种格式。

4. 第四种类型

行号　IF……GOSUB　‹子程序中第一条语句的行号›

R1微型机也不能引用这种格式。需要处理这种语句时，应将它改写成第二种类型的格式。即

行号　IF……THEN　GOSUB　‹子程序中第一条语句的行号›

四、应用举例

【例1】设计一段计算托运行李运费的程序。

已知：若重量$W\leqslant 50$kg，则每公斤运费为$x=0.13$元；若重量$W>50$kg，则超过50公斤的部分每公斤按0.20元计算。

依题意列出算式：

$$x=\begin{cases} W\times 0.13 & \text{当}W\leqslant 50\text{kg} \\ 50\times 0.13+(W-50)\times 0.20 & \text{当}W>50\text{kg} \end{cases}$$

计算程序如下：

```
10  INPUT  W
20  PRINT  "W="; W,
30  IF  W>50  THEN  GOTO  60
40  X=0.13*W
50  GOTO  70
60  X=0.13*50+(W-50)*0.20,
70  PRINT  "X="; X
80  GOTO  10
RUN  ENTER
W=34                X=4.42
W=58                X=8.1
W=50                X=6.5
```

值得注意的是：50行语句是不能缺少的，因为在30行语句中，如果判断W不大于50公斤时，则按40行语句的表达式计算，计算完毕，即按照50行GOTO语句的意思，转去执行70行打印语句。

程序也可以改写为

```
10  INPUT  W
20  PRINT  "W="; W,
30  IF  W<=50  THEN  X=0.13*W
40  IF  W>50  THEN  X=0.13*50+(W-50)*
0.20
50  PRINT  "X="; X
60  GOTO  10
```

【例 2 】编写一个打印$\sqrt{x}$值的程序。

根据题意列出算式：

$$y=\begin{cases}\sqrt{x} & \text{当 } x\geqslant 0\text{时}\\ j\sqrt{|x|} & \text{当 } x<0\text{时}\end{cases}$$

为方便计算起见，把上式改写为

$$y=\begin{cases}\sqrt{|x|} & x\geqslant 0\text{时}\\ j\sqrt{|x|} & x<0\text{时}\end{cases}$$

这样不管 X 是正值还是负值，都可以用一个算式 $Y=\sqrt{|X|}$ 来处理，而当X为负值时，再加印"j＊"字样。计算程序如下：

```
10  INPUT  X
20  Y=ABS  X
30  IF  X<0  THEN  PRINT  "J＊";
40  PRINT  SORY
50  GOTO  10
```

【例 3 】根据银行年利率编一个程序，计算若干年后利息可大于或等于本金，并分别打印出本金数、利率、年数和本息数。

设 P＝本金

R＝年利率

Y＝年数

Q＝本息（即本金＋利息）

依题意列出算式：

$$Q=P(1+R)^{Y}$$

计算步骤：求 Y＝1时的本息，Y＝2 时的本息，直至本息大于或等于原来本金的二倍时为止（用条件语句来判

断）。编写出程序如下：

```
10  INPUT  P
20  INPUT  R
30  Y=1
40  Q=P*(1+R)**Y
50  IF  Q>=2*P  THEN  GOTO  80
60  Y=Y+1
70  GOTO  40
80  PRINT  "P=";  P;  "␣␣";  "R=";  R;  "␣␣";  "Y=";  Y;  "␣␣";  "Q=";  Q
RUN  ENTER
打入  100  ENTER  （键盘输入本金100）
打入  0.05  ENTER  （键盘输入年利率5%）
P=100  R=0.05  Y=15  Q=207.89282
```

说明：本例是有意识地直接引用算式$Q=P(1+R)^Y$，写程序时虽然少费些功夫，但程序执行的时间却大大加长了。打入RUN命令，键入本金和年利率后，要等待几分钟才能显示出计算结果。

【例4】从键盘输入三个数，找出其中最大的一个数打印出来。

设输入三个数为A，B，C。它们有下列三种情况：

（1）A>B且A>C时，则A为最大；

（2）A>B且C>A时，则C为最大；

（3）A<B且B>C时，则B为最大。

应用条件语句，可以写出如下两段程序。

程序1：

```
10  INPUT  A
```

```
20  INPUT  B
30  INPUT  C
40  IFA>B THEN IF A>C THEN PRINT  A
50  IF A>B THEN IF C>A THEN PRINT  C
60  IF B>C THEN IF A<B THEN PRINT  B
```

程序2：

```
10  INPUT  A
20  INPUT  B
30  INPUT  C
40  IF  A>B  AND  A>C  THEN  PRINT  A
50  IF  A>B  AND  A<C  THEN  PRINT  C
60  IF  B>C  AND  A<B  THEN  PRINT  B
```

由上述两段程序可见，程序1采用的是条件语句再套一个条件语句，而程序2采用的是逻辑与条件的条件语句，两者效果是一样的。

习　　题

〔4-1〕　编一个求A^N的程序，其中N为大于1的整数，A为任意值。要求用INPUT语句输入A，N值。

〔4-2〕　编一个由键盘输入三个数，计算机自动将这三个数按大小次序打印出来的程序。

〔4-3〕　编一个程序：输入一组数据（均小于1000），找出其中最大的数。

〔4-4〕　编一个程序，计算

$$y=\begin{cases}10 & \text{当 } 0\leqslant x<10\text{时}\\ 20 & \text{当 } 10\leqslant x<20\text{时}\\ 30 & \text{当 } 20\leqslant x<30\text{时}\end{cases}$$

x 值由INPUT语句输入。

〔4-5〕 已知电容器放电公式为

$$U = U_0 e^{\frac{-t}{RC}},$$

求当 $t=1$ 秒时，把 $U \leqslant 0.05U_0$ 的那一组 R、C 的值打印出来。

数据						
数据	$R(\Omega)$	470K	50K	80K	63K	650K
	$C(\mu F)$	1	2	2.5	5	5

〔4-6〕 编写一个程序，计算 $T=\Sigma N^2$ 的值。式中 N 由1，2，3…变化，直到 T 等于或大于 10^4 为止。打印出 T 和 N 值。

第五章　如何编写程序

引入了转向(GOTO)语句、条件(IF-THEN)语句以及将要讨论的循环语句和转子语句后，我们可以利用它们去控制程序的转向，使得程序不必按行号的大小顺序运行。由于程序出现了分支，所以我们在处理比较复杂的问题时，容易被程序的走向弄得糊涂而产生错误。因此，有必要对BASIC程序的编写加以讨论。

第一节　编程的步骤

对比较复杂的问题，编写程序的步骤大致分为四步。

第一步：分析题意，确定解决问题的方案和计算方法。

第二步：画出执行任务的流程图（框图）。

第三步：按框图编写程序。

第四步：将写好的程序上机调试，边试边改，并通过数值验证，最后把程序确定下来。

在上述的步骤中，第一步是最关键的，因为程序里的许多问题都需要由这一步来确定。如采用几个变量，变量的初值如何，程序的结构如何，等等。

BASIC程序一般分为三个部分：

（1）输入数据和给变量赋初值；

（2）主体内容；

（3）将结果值输出打印。

第二节　框图（程序流程图）

用文字叙述计算过程很不直观，对复杂的程序往往不容易说清它们的结构以及各个部分之间的关系。因此人们常用一种图示方法（即框图表示方法），形象地、直观地表达出程序的设计思想、解题步骤和程序的走向。在框图的每个框内写明计算机要做的事情，框与框之间用箭头表示出程序的流程。每个框代表一个程序段（只包括一个或几个语句），使得在编程时能把复杂的程序化为几段简单的程序，检查程序时也比较容易从框图上发现程序设计是否有错。如果把语句标号注在对应的框图的旁边，这样便把框图与程序对应起来。所以框图可以帮助我们编写程序、检查程序、阅读程序和交流程序。

对简单的计算问题，画框图这一步骤可以省略。

框图中的每一框表示一个程序段的功能。一般一个框只需要一个语句或几个语句。常用的图形符号见图 5－1 所示。

现对几种主要的框图作详细的介绍。

矩形框表示计算机执行的叙述、处理、运算和加工等工作。计算公式通常直接写在矩形框中。

菱形框（判断框）具有判断的功能。它表示在两个可选择的路径中，判断出选取某一条路径，以便决定程序的转向。它与条件语句相对应，可以根据给定的条件，判断出所走的路径。菱形框有两个分支，即有一个入口、两个出口，其中一个出口是对应“满足”条件（通常用“是”表示）的分支，另一个出口是对应“不满足”条件（用“否”表示）的分支。

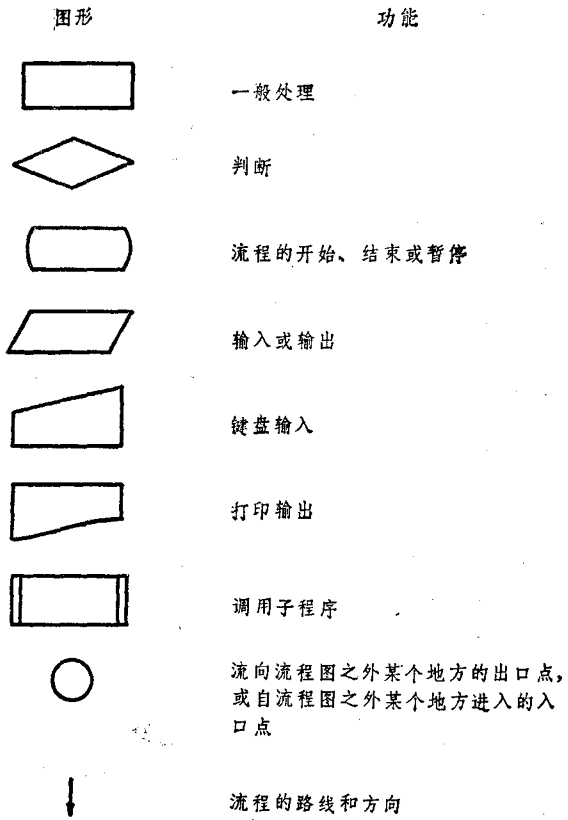


图5·1　常用的图形符号

箭头表示程序执行的流向。

画流程图时也可将框省略不画，只留下箭头和框中的说明。至于框中说明的详细程度，可根据要求而定，有较粗略表示程序总体结构的“粗框图”，也有表示程序详细结构的“细框图”。我们可以先画出“粗框图”，然后再画出“细框

图”。“细框图”基本上能反映出程序所用的语句。

下面举例说明框图的作用。

【例1】求二次方程式

$$Ax^2+Bx+C=0$$

的根。

1. 确定算式

（1）当$B^2-4AC\geqslant 0$时，方程有两个实根：

$$x_{1,2}=\frac{-B\pm\sqrt{B^2-4AC}}{2A},$$

（2）当$B^2-4AC<0$时，方程有两个复根：

$$x_{1,2}=-\frac{B}{2A}\pm i\frac{\sqrt{-(B^2-4AC)}}{2A}。$$

由上两式看出：计算方程式的根，首先要判断B^2-4AC是否小于0，然后才能确定采用的算式。

2. 画出框图

如图5-2所示。

3. 根据框图编写程序

图5-2中框图的右旁括弧内的数字，表示该框对应于程序中的行号。从框图可分析出各个框应采用的语句。即：

框①用键盘输入语句比较合适。

框②、④、⑤、⑥、⑦要用赋值语句。

框③用条件语句。

框⑤和⑥用打印语句。

框⑨要用暂停语句。

经过以上的分析，不难写出如下程序：

```
10  INPUT  A
20  INPUT  B
```

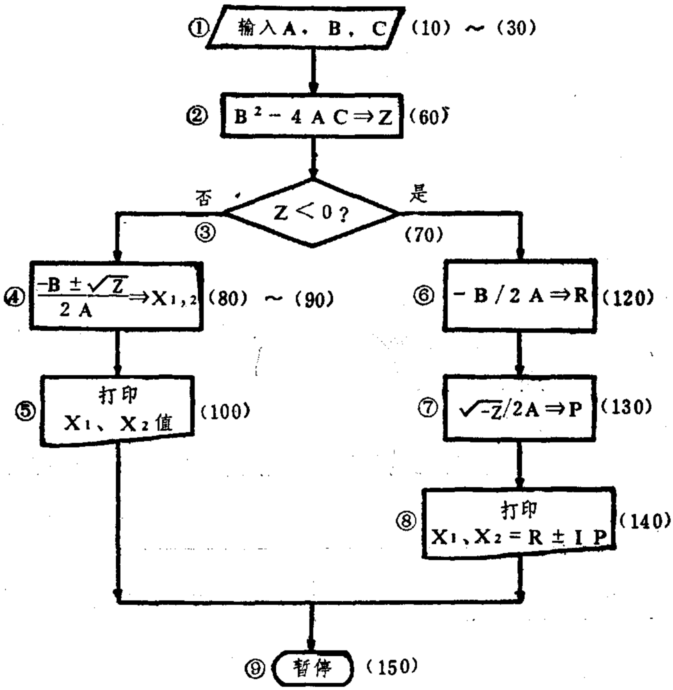


图5-2

```
30  INPUT  C
40  PRINT  "A＝"; A; "␣␣"; "B＝"; B;
"␣␣"; "C＝"; C
50  PRINT
60  Z＝B＊B－4＊A＊C
70  IF  Z＜0  THEN  GOTO  120
80  X1＝(－B＋SQR  Z)／(2＊A)
90  X2＝(－B－SQR  Z)/(2＊A)
100  PRINT  "X1＝"; X1, "X2＝"; X2
```

```
110  GOTO  150
120  R = - B/(2 * A)
130  P = SQR( - Z)/(2 * A)
140  PRINT  "X1 = "; R; " + I"; P, "X2 = "; R; " - I"; P
150  STOP
```

【例 2 】已知二十个车间的月产值，现要求用列表的形式将产值分为四类：①大于 1 万元的；② 5 千元到 1 万元的；③ 1 千元到 5 千元的；④不大于 1 千元。并分别打印出属于各类产值的车间总数。

1．分析题意

根据题意列出需要打印的产值分类表，如表 5 - 1 。

表5-1

车间 \ 产值X \ 分类	X>10000	5000<X≤10000	1000<X≤5000	X≤1000
1 车间 ⋮ 20车间				
各类车间总数	A	B	C	D

在本题中，程序的编写可从下述两方面去考虑：

（ 1 ）输入X值，并判断X值所属的类别，然后在该类的栏内打印出X值，并累计每类共有的车间总数。因此需要用到五个变量，即各车间月产值 X，第一、二、三、四类共有的车间总数A，B，C，D。

累计公式为

A＝A＋1，B＝B＋1，

C＝C＋1，D＝D＋1。

从算式中可知：必须事先用0赋给变量A，B，C，D。

（2）判断输入的X值是否结束，如果"是"，则转去打印各类车间总数。这里采用办法为：用一个变量K来累计输入次数，在K不超出终值的范围内，每输入一个X值，将K值加1。

2．画出框图（见图5－3）

框图的次序为

打印表头──→给变量赋初值和输入X值──→主体内容（分类打印产值，累计各类车间数）──→输出结果

3．根据框图编写程序

```
10  PRINT TAB(4); "1"; TAB(12); "2"; TAB(20); "3"; TAB(28); "4"
20  PRINT
30  K=0
40  A=0
50  B=0
60  C=0
70  D=0
80  IF K>20  THEN  GOTO  260
90  INPUT  X
100  K=K+1
110  IF  X>10000  THEN  GOTO  170
120  IF  X>5000  THEN  GOTO  200
130  IF  X>1000  THEN  GOTO  230
140  PRINT  TAB(26); X
150  D=D+1
```

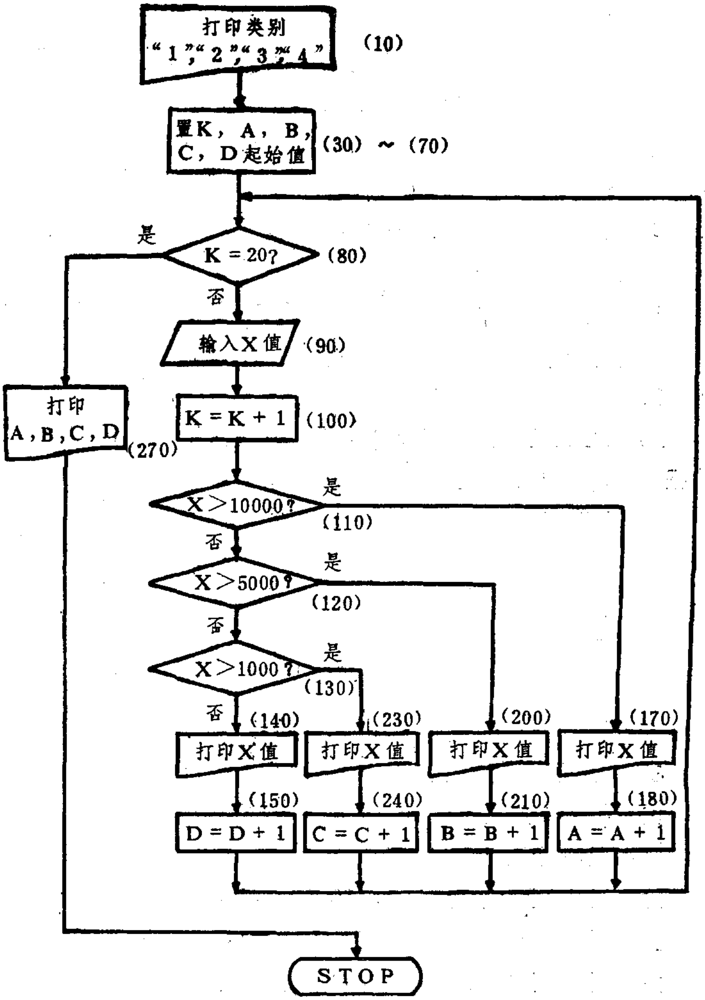

打印类别
"1","2","3","4"
(10)
置K，A，B，C，D起始值
(30) ~ (70)
K = 20?
(80)
是
否
输入X值
(90)
K = K + 1
(100)
打印
A，B，C，D
(270)
X > 10000?
(110)
X > 5000?
(120)
X > 1000?
(130)
(140)
(230)
(200)
(170)
打印X值
(150)
(240)
(210)
(180)
D = D + 1
C = C + 1
B = B + 1
A = A + 1
STOP

图5-3

```
160  GOTO  80
170  PRINT  TAB ( 2 ) ; X
180  A = A + 1
190  GOTO  80
200  PRINT  TAB (10) ; X
210  B = B + 1
220  GOTO  80
230  PRINT  TAB (18) ; X
240  C = C + 1
250  GOTO  80
260  PRINT
270  PRINT  TAB ( 2 ) ; "A = "; A; TAB(10);
"B = "; B; TAB (18) ; "C = "; C; TAB(26); "D = "; D
280  STOP
```

由于R1微型机的每个显示行只有32格，所以在编写这个程序时，无法打印出各个类别的产值范围，因而改用1，2，3，4代表各类别。打印语句中引用了TAB（X）函数，此函数的含义是：打印的字头打在X值所指的位置上。如10行的打印语句是表示在第4格的位置上打印1，在第12格的位置上打印2，在第20格的位置上打印3，在第28格的位置上打印4。

程序中行号标在框图旁，以便于框图与程序之间的对照阅读。

第三节　程序的优化考虑

在同一设计要求的程序中，可以用各种不同的方式去编

写。那么，是不是在程序中语句越少越好呢？如何衡量一个程序的优劣呢？一般来说，一个程序的好坏主要从两方面去考虑：时间和空间，即程序运行所需时间及所占用的内存空间，这两个因素既是密切联系的，又往往是相互矛盾的。

我们现在来讨论一下如何把一个程序编写好。

一、尽量减少重复计算

【例3】计算$5tg^3x+6tg^2x+7tgx+8$的值。

```
10  INPUT  X
20  PRINT  5*(TANX)**3+6*(TANX)**2
+7*TANX+8
```

乍看起来，这个程序很简单，只用了两个语句。但仔细分析一下，可发现在打印语句中，TANX需多次运算，浪费了机器的运行时间。

我们把程序改动如下：

```
10  INPUT  X
20  A=TANX
30  PRINT  5*A**3+6*A**2+7*A+8
```

这一程序多用了一个赋值语句，但较前一个程序优越，因为它只需计算一次TANX的值，能够缩短程序执行时间。但这个程序也很不完善，如果把$5A^3+6A^2+7A+8$用多项式的方法（见下例）来处理，就可得到较好的程序。表面看来，这个程序所含语句比前两个程序多，但程序的运行时间可大大缩短。

二、写成统一算式

先将算式统一，再利用条件语句、GOTO语句和循环语句使程序重复运行。如程序中出现有重复使用的程序段，则写成子程序（详见第六章），以节约所占用的内存单元。

【例 4】计算多项式

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$的值。

为便于说明起见，先以三次多项式为例。

$$Y = AX^3 + BX^2 + C。$$

具体计算方法如下：

1. 分析题意，写出算式

编写这个程序时，可以采用例 1 的方法来处理，即输入 A，B，C，D，X 值及用一个赋值语句。采用例 1 的方法编写的程序虽然很简单，但 X 的幂次运算要花较多机器时间。此外，当最高次幂不同时，还得给程序加以修改，增加一些语句，所以采用例 1 的方法编程不是最优的。

如果把算式改为

$$Y = ((AX + B)X + C)X + D,$$

则求 Y 值可用统一算式

$$Y = YX + J。$$

第一次运算时，首先把最高次幂的系数A赋给变量 Y，或者说变量 Y 的起始值等于最高次幂的系数。即

$$A \Longrightarrow Y, \qquad B \Longrightarrow J。$$

然后计算出YX + J之值后赋给变量 Y：

$$YX + J \Longrightarrow Y$$

第二次运算时，

$$C \Longrightarrow J, \qquad YX + J \Longrightarrow Y。$$

后一赋值式中左边 Y 值是上次运算的结果，它的值为 AX + B，因此第二次运算相当于计算了(AX + B)X + C。

第三次运算时，

$$D \Longrightarrow J, \qquad YX + J \Longrightarrow Y。$$

这次赋值相当于计算：

$$Y = ((AX + B)X + C)X + D。$$

由此得出以下几点结论：

(1) Y的起始值等于最高次幂的系数。而J值则从较高次幂的系数开始依次赋值，直到常数项为止。

(2)如最高次幂为N时，需进行N次Y = YX + J运算。

(3)化为统一算式Y = YX + J是为了便于循环使用，减少程序的重复编写，节约内存空间。同时这个算式亦很简单，可以缩短执行时间。

2. 画出框值（见图5-4）

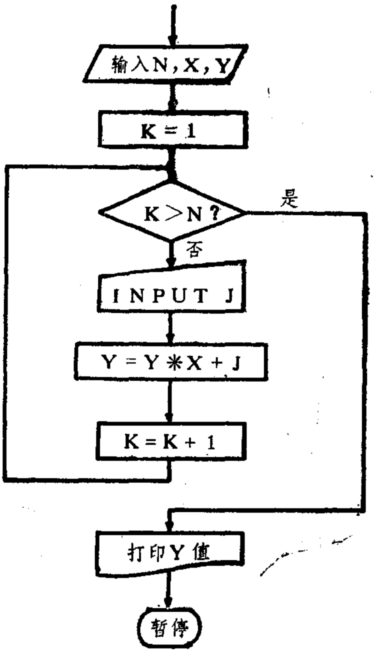


图5-4

3. 根据框图写出程序

```
10 INPUT N
20 INPUT X
30 INPUT Y
40 K=1
50 IF K>N THEN GOTO 100
60 INPUT J
70 Y=Y*X+J
80 K=K+1
90 GOTO 50
100 PRINT Y
110 STOP
```

三、简化程序

简化程序就是把不必要的行或语句消去。如充分利用允许使用多语句的程序行，则应尽量少占行数，因为输入一个新的行号也要占用内存单元。

【例5】从输入的数据中将正数挑选出来打印。

本例可编写出如下几种程序。

程序1：

```
10 INPUT X
20 IF X>=0 THEN GOTO 40
30 GOTO 10
40 PRINT X
50 GOTO 10
```

程序2：

```
10 INPUT X
20 IF X<0 THEN GOTO 10
30 PRINT X
```

```
40  GOTO  10
```

程序3：

```
10  INPUT  X
20  IF  X>=0  THEN  PRINT  X
30  GOTO  10
```

比较以上三个程序，无疑采用程序3是较优方案，因为这三个程序所用语句基本相同，而程序3的行数最少，可节约内存单元。

四、用连乘的运算代替幂运算

当n为整数时，最好用连乘的方法来计算x^n的值。如将x^2写成X＊X，而不写成X＊＊2，这样可节省计算机的执行时间。

现以第四章例3的计算银行本息为例予以说明。当时是直接引用算式$Q=P\times(1+R)^Y$进行运算的，因为每次计算都要进行幂运算，所以程序执行时间大大加长了。现将运算的方法作如下修改：

设开始时存入银行本金为P0。第一年本息的计算方法为

P0⟹P

P＊(1＋R)⟹P　（作为第二年本金）

以后每年的本息计算方法为

P＊（1＋R）⟹P

当P/P0>2时运算结束。

编写程序如下：

```
10  INPUT  P0
20  INPUT  R
30  P=P0
40  Y=1
```

```
50  P=P*(1+R)
60  IF  P/P0>=2  THEN  GOTO  90
70  Y=Y+1
80  GOTO  50
90  PRINT  "P0=";  P0;  "␣␣";  "R=";  R;
"␣␣";  "Y=";  Y;  "␣␣";"P=";P
RUN  (ENTER)
打入  100  (ENTER)      （键盘输入本金100）
打入  0.05  (ENTER)   （键盘输入年利率5%）
P0=100  R=0.05  Y=15  P=207.89282
```

执行这个程序时，只需重复计算P*(1+R)的简单式子，使得程序的执行时间大大地缩短。

五、程序中尽量压缩注释语句的篇幅

因为将REM语句编入程序的作用仅仅为了说明，机器并不作任何运算，所以减少使用注释语句，不但不会影响计算的结果，而且可以节省内存空间。

此外，在使用说明语句（DIM）时，要准确地定义下标变量的最大下标值和字符串变量的最大长度，以充分利用内存空间。

写表达式和函数时，尽量少用圆括号。这是因为处理圆括号时，计算机要占用较多的字节和时间。

如果中间结果不必打印的话，则应省去相应的打印语句，这样也可节省内存空间和时间。

第四节　注释语句

在编写程序时，往往需要在程序段的开始前加上一段注

释性的文字。要实现这一目的，就要引入注释语句。

注释语句的格式是

REM（‹注释内容›）

REM是REMARK（注释）一词的缩写。它的作用只是对程序作注释，相当于给程序段写个标题或取个名字，它可以用来注明程序段的内容、作用、计算方法和步骤等，以便于阅读程序和检查程序。例如：

100 REM SUBROUTINE TO CALCULATE THE N FACTORIAL

这个注释语句用来注明一个求N阶乘的子程序。又如：

100 REM* *MAIN PROGRAM* *可以用来注明"主程序"。又如：

```
1000 REM PROGRAM TO DRAW SQUARE
1010 REM (X, Y)=CENTER S=SIZE
```

以上两句分别注明一个画正方形的程序和以点（X，Y）为中心、以中心到每边的距离为S。这两个语句可以提醒人们在调用这段程序时，应事先对变量X，Y和S赋值。

REM语句和DATA语句一样，是一个非执行语句。计算机在遇到这个语句时，不执行任何操作。当输入LIST命令时，计算机会将REM语句中的字符全部按原样印出来，但在输入RUN命令时，计算机不做任何工作。

第六章　程序控制的有关语句

第一节　循 环 语 句

一、问题的提出

在BASIC程序设计中，经常会遇到一些按一定的规律重复多次运算的问题。例如：求1到100之间各整数的平方值，即计算1^2，2^2，3^2，…，100^2，需要重复100次的平方运算。又例如：每次打印一个星号"*"，要求打印100次。在这里，虽然重复的内容与上例不相同（是打印而不是计算），但也需要统计打印次数，每打印一次，统计一次，同样需要重复100次的运算。现将以上两例用赋值语句来编写程序。

程序1：

```
10   X = 1     （初值）
20   T = X * * X
30   X = X + 1    （增量为1）
40   T = X * * X
 ⋮
1900   X = X + 1    （终值）
2000   T = X * * X
```

程序2：

```
10   X = 1    （初值）
20   PRINT  "*"
30   X = X + 1    （增量为1）
```

```
40  PRINT  "*"
 ⋮
```

程序1是一个计算1到100的范围内整数的平方值的程序。程序2是一个打印100个星号的程序。显然这样编出的程序是十分冗长的。但是我们可以由此得到启发：首先要找出其中的规律，写出统一的算式，然后用一个变量去控制重复的次数，此时必须知道变量的三个参数——初值、增量值和终值。第一次变量按初值进行计算，以后每次加上增量值，然后判断它是否超过了终值，如果没有超过，则重复进行下一次的计算。据此，我们可用条件语句将程序1改写为

```
10  X=1
20  T=X**X
30  X=X+1
40  IF  X<100  THEN  GOTO  20
50  STOP
```

在这个程序中，10行的内容是给变量X赋初值，20行的内容是计算，30行和40行的内容是将运算后的变量加上增量值，接着判断它是否超过终值，如果没有超过终值，则转到第20行继续运算，直到变量X超过终值时程序暂停运行。

现用循环语句(FOR-NEXT语句)将程序1编写如下：

```
10  FOR  X=1  TO  100  STEP  1
20  T=X**X
30  NEXT  X
40  STOP
```

在这个程序中，FOR-NEXT语句包含了如下内容：给变量X赋初值，每次运算后加步长值（增量）和判断X值是否超过终值。这个语句与条件语句比较起来有不少优点，优

Retro Workshop
果粉工作室

点之一是初值、终值和步长值一目了然，循环次数容易算出。

二、循环语句的一般格式

循环语句的一般格式为

```
FOR ‹循环变量›=‹初值›  TO  ‹终值›  STEP
‹步长值›
〈循环体〉
NEXT〈循环变量〉
```

这一段程序的含意是：循环变量的取值由初值开始到终值结束，每次间隔为步长值，循环变量每取一个值，循环体就被执行一次。

使用循环语句时几项具体规定：

（1）FOR-NEXT语句必须成对出现。FOR语句称为循环起始语句，NEXT语句称为循环终端语句。

（2）FOR和NEXT语句中所用的变量必须相同，这个变量称为循环变量。

（3）循环变量必须是简单变量。

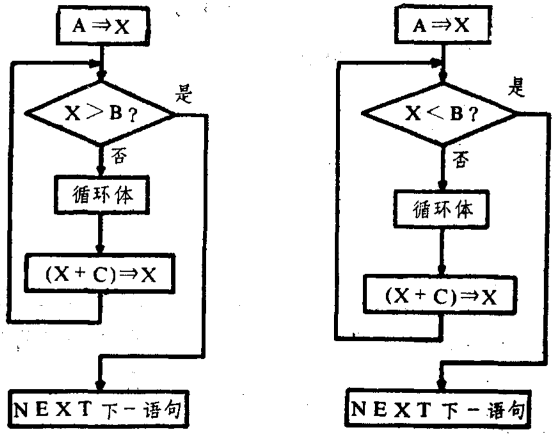
（4）初值、终值和步长值可以是具体数值，可以是已赋值的变量，也可以是表达式，但表达式所用变量必须是已赋值的。

（5）步长值可以是正值，也可以是负值。当步长为负时，循环变量必须小于终值才可能停止循环。当步长值为1时，“STEP”和步长值可以省略。

三、循环语句的执行过程

在图6－1中，X为循环变量，A为循环变量初值，B为循环变量终值，C为步长值。

下面举一个简单的循环程序，并结合图6－1所示的框



（a）步长为正值　　　（b）步长为负值

图6-1　循环语句的程序框图

图，来说明循环语句的执行过程。

【例1】

```
10  FOR  X=1  TO  10  STEP  1
20  PRINT  X;  "␣";
30  NEXT  X
40  STOP
RUN
1  2  3  4  5  6  7  8  9  10
```

这个程序的执行情况是这样的：执行10行的FOR语句时，先给变量X赋初值，然后判断X是否超过终值；执行20行循环体的语句时，打印出X值；执行30行的NEXT语句时，将X值加上步长值。在进行第一次循环时，X=1；在进行第二次循环时，X=2；……一直进行到第11次循环，此时X=11，X值超过了终值，于是转到执行40行的语句，即

暂停运行。整个循环过程如表6-1所示。

表6-1

循环次数	循环过程（括号内数值表示运行的行号）		
1	(10) X = 1　X>10?	(20)打印 1	(30)X = 2
2	(10)X = 2　X>10?	(20)打印 2	(30)X = 3
3	(10)X = 3　X>10?	(20)打印 3	(30)X = 4
4	(10)X = 4　X>10?	(20)打印 4	(30)X = 5
5	(10)X = 5　X>10?	(20)打印 5	(30)X = 6
6	(10)X = 6　X>10?	(20)打印 6	(30)X = 7
7	(10)X = 7　X>10?	(20)打印 7	(30)X = 8
8	(10)X = 8　X>10?	(20)打印 8	(30)X = 9
9	(10)X = 9　X>10?	(20)打印 9	(30)X = 10
10	(10)X = 10　X>10?	(20)打印 10	(30)X = 11
11	(10)X = 11　X>10?	(40) 暂停	

【例 2】

```
10  FOR  X=10  TO  1  STEP  -1
20  PRINT  X;  "␣";
30  NEXT  X
40  STOP
RUN
10  9  8  7  6  5  4  3  2  1
```

在例 1 中，步长为正值，X值是由 1 到10递增变化，当循环变量超过终值（X>10）时结束循环。在例 2 中，步长

为负值，X值由10到1递减变化，递减到0时超过了终值而结束循环。

循环次数的计算公式为

$$循环次数=\frac{终值-初值}{步长}+1 \quad （取整数）$$

如：例1的循环次数 $=\frac{10-1}{1}+1=10$

例2的循环次数 $=\frac{1-10}{-1}+1=10$

循环体内的变量允许出现循环变量（如例2中的X），此时初值、终值和步长值不能随意选定，它们的数值与循环体的内容有关。循环变量也可以不在循环体内出现，此时循环变量只是用来控制循环的次数，所用变量的名字、初值、终值和步长值可以随意选定。

【例3】

```
10  X=5
20  FOR  I=1  TO  10  STEP  2
30  PRINT  X;
40  NEXT  I
RUN
 5   5   5   5   5
```

这个程序的循环语句可改为

```
20  FOR  I=11  TO  20  STEP  2
40  NEXT  I
```

也可改为

```
20  FOR  J=1  TO  5
40  NEXT  J
```

在使用循环语句时，不允许通过条件语句或转向语句从循环体外不经FOR语句而直接转入循环体内。如以下的程序是错误的：

```
10  GOTO  30
20  FOR  I=1  TO  5
30  PRINT  I
40  NEXT  I
```

此程序的执行过程是由10行转到30行，跳过了20行语句，这时由于I未赋值。因此在执行30行语句时，无法将I值打印出来。

在使用循环语句时，允许从循环体内不经NEXT语句而直接转出循环体。因为在某些情况下，程序不必将整个循环全部执行。以下的程序是正确的：

```
10  FOR  I=1  TO  5
20  IF  I=3  THEN  GOTO  50
30  PRINT  I
40  NEXT  I
50  STOP
```

四、循环语句的举例

编写程序时，首先应分析题意，写出计算公式。要完成这一步往往是比较困难的，下面通过一些例子来说明编程的思路。这里，我们必须注意赋值语句（算式）的使用、变量的个数、变量的初值和语句的选用。

【例4】求$S=\sum_{n=1}^{10} n$的值。

1．分析题意

$$S=\sum_{n=1}^{10} n=1+2+3+\cdots+10$$

本题的计算与在算盘上做累加运算很相似。

运算前先将变量清零，即

$S=0$；

第一次加1，得

$S=0+1$；

第二次在原值上加2，得

$S=(0+1)+2$；

第三次又在原值上加3，得

$S=(0+1+2)+3$。

由此可找出其运算规律：S的新值等于S的原值加上X（X的取值范围是由1到10，每次取值递增1）。

因此得赋值式：

$S=S+X$。

2. 写出程序中语句次序

对简单而语句不多的程序，不一定都要先画框图。我们可以根据上述思路，拟出一个大概的步骤：

给X赋初值⟶FOR语句⟶循环体S=S+X⟶NEXT语句⟶输出打印

3. 写出程序

```
10  S=0
20  FOR  X=1  TO  10
30  S=S+X
40  NEXT  X
50  PRINT  "S=";S
```

【例5】求T＝20！的值。

1．本题是连乘运算，编程思路与例4十分类似，但T的起始值应为1而不是0。

第一次做乘1运算，即

$T=1\times1$；

第二次在原值上再乘以2，得

$T=(1\times1)\times2$。

由此得赋值式：

T＝T＊X，

式中X值由1取到20，步长为1。

由上述分析可知，本题适用循环语句来编程。

2、写出程序中各语句的次序

给T赋初值──→FOR──→循环体T＝T＊X──→NEXT──→打印输出

3．写出程序

```
10  T=1
20  FOR  X=1  TO  10
30  T=T*X
40  NEXT  X
50  PRINT  "T=";  X
```

【例6】求$1+2^1+2^2+2^3+\cdots+2^{10}$的值。

1．先求每一项之值，并用变量T表示：

T＝2＊＊X，

其中，X为循环变量，X值由0取到10，步长为1。

2．求累加和，用变量S表示：

S＝S＋T，

其中，S的初值为0。

3. 写出程序

```
10  S=0
20  FOR  X=0  TO  10
30  T=2**X
40  S=S+T
50  NEXT  X
60  PRINT  S
```

【例 7】用循环语句写出计算n次多项式

$$a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$$

之值的程序（参阅第五章例 4 的程序）。

```
10  INPUT  N
20  INPUT  X
30  INPUT  Y
40  FOR  I=1  TO  N
50  INPUT  J
60  Y=Y*X+J
70  NEXT  I
80  PRINT  Y
90  STOP
```

因R1微型机没有READ/DATA语句，所以在此程序中，只能用INPUT语句输入Y值和J值。如计算某一具体多项式时，使用READ/DATA语句会方便些。

例如：计算

$$5.5x^5+4.4x^4+3.3x^3+2.2x^2+1.1x+0.9$$

在$x=5$时的值，可编写程序如下：

```
10  READ  X
20  READ  Y
```

```
30  FOR  I=1  TO  5
40  READ  J
50  Y=Y*X+J
60  NEXT  I
70  PRINT  Y
80  DATA  5, 5.5, 4.4, 3.3, 2.2, 1.1, 0.9
90  STOP
```

【例8】已知一个数列，它的第一个数为0，第二个数仍为0，第三个数为1，以后每个数都是前三个数之和。试编写一个求这数列前30个数的程序。

通过分析题意，可列出每次求得的三个数值A，B，C，并列表如下：

X	A	B	C
1	0	0	1
2	1	2	4
3	7	13	24
4	44	81	149
⋮	⋮	⋮	⋮
10	2555757	4770770	8646064

应用赋值式

$$A+B+C\Longrightarrow A$$

$$B+C+A\Longrightarrow B$$

$$C+A+B\Longrightarrow C$$

可编得程序如下：

```
10  A=0
20  B=0
30  C=1
```

```
40  FOR  X=1  TO  10
50  PRINT  TAB(5); X; TAB(10); A; TAB
(15); B; TAB(20); C
60  A=A+B+C
70  B=B+C+A
80  C=C+A+B
90  NEXT  X
100  STOP
```

程序的50行引用了TAB(X)函数(详见第九章)，规定了从每行的第5格、第10格、第15格和第20格上开始分别打印出X，A，B和C值。

五、多重循环

1. 多重循环的概念

我们来看看一个打印"九九表"的例子。

"九九表"的内容为

$$
\begin{array}{llll}
1\times1=1 & 1\times2=2 & \cdots & 1\times9=9 \\
2\times1=2 & 2\times2=4 & \cdots & 2\times9=18 \\
\vdots & \vdots & & \vdots \\
9\times1=9 & 9\times2=18 & \cdots & 9\times9=81
\end{array}
$$

其中每一项的通式为

P=A*B

当A=1时，B值取遍1，2，…，9，即执行一次变量B的循环；当A=2时，又执行一次变量B的循环；……直到A=9时为止。这时A值也是取遍1，2，…，9，所以A也是一个循环变量。由于每给定一个A值，就需要变量B作一次循环，或者说，B循环被套在A循环里面。我们把这种结构称为循环的嵌套，把变量B的循环称为内循环，而变量A的

循环称为外循环。

2．框图

内循环框图是一个单循环：

给B赋初值──→判断B是否超过终值──→执行循环体──→加步长后作循环，转到比较框，直到B>9时脱离循环。

如把内循环看作是外循环中的循环体，则外循环也完全符合循环框图的结构。内循环完全被套在外循环里面。

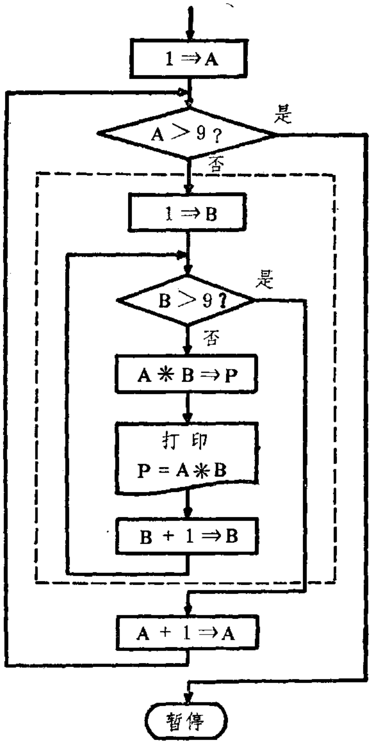


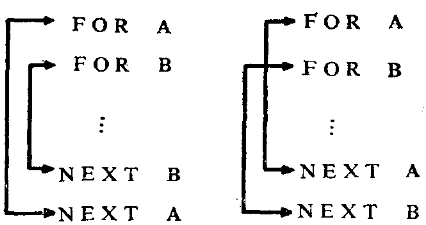
图6-2

这个双重循环的执行情况是这样的：当A＝1时，B以步长为1的变化规律从1递增到9，B每取一个值就执行一次A＊B⟹P，于是打印出1×1＝1到1×9＝9。内循环执行完毕（B>9）后，程序执行A＋1⟹A，此时A＝2。由于A<9，程序又重新执行内循环的循环体内容。如此重复进行，A值由1依次变到9，打印出9×1＝9到9×9＝81后，外循环结束，程序的执行暂停。

```
10  FOR  A=1  TO  9
20  FOR  B=1  TO  9
30  P=A*B
40  PRINT  A; "*"; B; "="; P; "␣␣";
50  NEXT  B
60  PRINT
70  NEXT  A
80  STOP
```

4．多重循环的一些规定

在BASIC系统中，循环语句允许多层嵌套，但必须注意内外循环的层次要分明，不允许交叉。图6－3（a）中循环语句的安排是正确的，内循环被套在外循环里面，循环的



（a）　　　（b）

图6-3　双重循环

次序是先作内层循环，内循环结束后再转到外循环。图6-3（b）中循环语句的安排是错误的，它的内外循环层次互相交叉。

第二节 转子语句和返回语句

我们已经介绍了一些GOTO语句、IF-THEN语句和FOR-NEXT语句等程序控制语句，它们的作用都是把程序转到某一指定的行号并继续执行程序。而本节内容主要是讨论把程序转去执行需要多次重复使用的程序段。

一、什么叫子程序

我们首先看一个例子。

【例9】编出计算 S=2! +3! +7! 的程序。

在上一节里，我们已讨论了求N!和累加的程序，因而很自然地会将求S的程序编写如下：

```
10  S=0
20  N=2
30  P=1
31  FOR  J=1  TO  N
32  P=P*J
33  NEXT  J
40  S=S+P
50  N=3
60  P=1
61  FOR  J=1  TO  N
62  P=P*J
63  NEXT  J
```

```
70  S=S+P
80  N= 7
90  P=1
91  FOR  J=1  TO  N
92  P=P*J
93  NEXT  J
100  S=S+P
110  PRINT  S
```

这里，求N！的程序包括如下四个语句：

```
310  P= 1
320  FOR  J= 1  TO  N
330  P=P*J
340  NEXT  J
```

由这个程序可以看出，在每次求N！时都需要重复地编写出四行语句，这种写法十分烦琐，并且占用较多的内存单元。能否象初等数学那样，把一些常用的函数值事先列成一个函数表呢？如果有可能的话，在需要应用时就可通过查表来求出函数值，而不必每次都进行函数运算了。

在BASIC中，编写程序同样可以应用上述方法来处理。在一个程序中，如果需要多次重复使用一段程序，就可以把它编成一个相对独立的程序，并称它为子程序。需要使用这段程序时，可以转去执行这段子程序（称为调用子程序）。当子程序执行完毕后，就自动返回到原来的程序（称为主程序）中去，如图6-4所示。在主程序中，标号90行的内容转到子程序，子程序的最后一个语句（300行的语句）表明程序的执行返回到主程序中转子语句的下一个语句（100行的语句）。

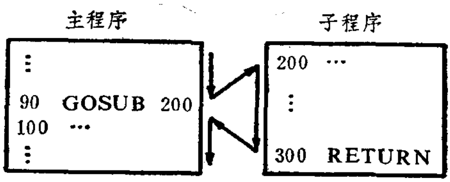


图6-4

二、转子语句和返回语句的格式

转子语句（GOSUB语句）的一般格式为

GOSUB‹子程序第一条语句的行号›

返回语句（RETURN语句）的一般格式为

RETURN

返回语句是作为子程序的最后一条语句。

现在可用转子语句和返回语句把例9写成如下程序：

```
10  S= 0
20  N= 2
30  GOSUB  300
40  S=S+P
50  N= 3
60  GOSUB  300
70  S=S+P
80  N= 7
90  GOSUB  300
100  S=S+P
110  PRINT  S
120  STOP
```

```
300  REM  SUB-N!   （求N! 子程序）
310  P = 1
320  FOR  J = 1  TO  N
330  P = P * J
340  NEXT  J
350  RETURN
```

现应用循环语句把这个程序的10～120行进行改写，可得到更为简短的程序：

```
10  S = 0
20  FOR  I = 1  TO  3
30  INPUT  N
40  GOSUB  300
50  S = S + P
60  NEXT  I
70  STOP
```

在编写这个程序时，需要注意如下两点：一是主程序中的最后一个语句STOP不能省略，否则就会发生主程序擅自进入子程序的错误；二是调用子程序前，在主程序中必须事先给子程序中所有未定义的变量(如本例的N)赋值，然后才能调用子程序。所以我们说，子程序只是具有相对的独立性。

三、子程序嵌套

子程序不能调用自己本身。但子程序可以调用另一个子程序，这种结构称为子程序的嵌套。子程序的嵌套必须层次分明，嵌套的次数不宜过多。子程序的嵌套如图 6 - 5 所示。

综合以上所述，调用子程序的规则如下：

1. 调用子程序时，必须在调用的地方写一个 GOSUB 语句。不允许采用其它方式进入子程序。

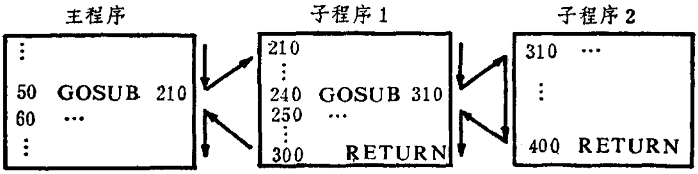


图6-5　子程序嵌套示意图

2. 子程序至少应有一个RETURN语句，当程序执行到此语句时，自动从子程序返回到主程序中调用本子程序的转子语句的下一个语句，继续执行主程序。不允许采用其它方式转出子程序。

3. 转子语句和返回语句必须成对出现。

4. 子程序可以嵌套，允许嵌套的层次要看具体的机型而定，使用者可参阅自己所用计算机的BASIC语句使用手册。

第三节　暂停语句

暂停语句（STOP语句）的作用是使程序暂停执行。其格式为

STOP

一个程序可以包含几个STOP语句。当程序执行到STOP语句时，程序就自动暂停并打印出如下的信息：

ST　IN　‹行号›

其中，行号表示STOP语句所在的行号，这个信息的含意是：计算机向使用者报告程序暂停的地方。如果需要重新执行程序，可以打入

GOTO 〈行号〉

的命令，其中行号指的是STOP语句的下一个语句的行号。此时程序就从STOP语句的下一个语句继续执行。

STOP语句可以用在程序的任何地方，一般是出现在某一程序段之末、子程序开始之前或主程序的结束之处。利用暂停语句使程序暂停执行，从而可以分段调试程序、检查程序和修改程序。

STOP语句的功能很象一个END语句（结束语句，R1机BASIC无此语句），但又与END语句有所区别。执行END语句时，一般的BASIC系统都会将程序中所有变量清零，而STOP语句只是命令计算机暂停执行，并不会将变量清零。

习　题

〔6－1〕 写出下面程序的执行结果（注意：在以后的书写中，用问号代替PRINT，用↙代替ENTER）：

(1)
```
10 FOR I=1 TO 4
20 ? I;
30 NEXT I
```

(2)
```
10 FOR J=-1 TO -10 STEP -3
20 X=J+1
30 ? X,
40 NEXT J
```

(3)
```
10 X=1
20 FOR T=1 TO 5
30 Y=T**(X+T)
```

```
40  IF  Y>20  THEN  GOTO  70
50  ? Y,
60  NEXT  T
70  ?  "Y>20"
```

〔6－2〕 以下程序运行完毕后，I的值是多少？

```
(1)10  FOR  I=2  TO  10  STEP  3
   20  ? I
   30  NEXT  I
(2)10  FOR  I =3.5  TO  5  STEP 1.5
   20  ?  I
   30  NEXT  I
(3)10  FOR  I=－1  TO  －5  STEP  －8
   20  ? I
   30  NEXT  I
```

〔6－3〕 以下的BASIC程序是否正确？为什么？

```
(1)10  FOR  I=1  TO  5
   20  FOR  J=1  TO  10
   30  ? I;
   40  NEXT  I
   50  ? J
   60  NEXT  J
(2)10  A=1
   20  FOR  I=－1  TO  －10  STEP  －1
   30  ?  Y=A+I
   40  IF  Y<=－2  THEN  GOTO 60
   50  NEXT  I
   60  STOP
```

〔6-4〕 写出下面程序的执行结果，并比较GOTO语句和GOSUB-RETURN语句的功能(设X=3，Y=2)：

```
(1)10  INPUT  X
   15  ?   "X=";  X,
   20  INPUT  Y
   25  ?   "Y=";  Y
   30  GOSUB  100
   40  ?   "T=";  T
   45  STOP
   100  Z=X**Y
   110  T=COS(Z*π/180)
   120  ?   "Z=";   Z,
   130  RETURN
(2)10  INPUT  X
   15  ?   "X=";  X,
   20  INPUT  Y
   25  ?   "Y=";   Y
   30  GOTO  100
   40  ?   T
   45  STOP
   100  Z=X**Y
   110  T=COS(Z*π/180)
   120  ?   "Z=";   Z
```

〔6-5〕 给出任意50个数，编写打印其中最大数和最小数的程序(50个数由键盘输入，有正、负数)。

〔6-6〕 用循环语句，编写求

$$\sum_{i=0}^{63} 3^i = 1 + 3 + 3^2 + \cdots + 3^{63}$$

的值的程序。

〔6－7〕 编写一个程序，用INPUT语句从键盘输入N个数据，当输入完毕后打印出数据个数以及数的总和。

〔6－8〕 设n从1到10每隔1取一值，编写程序造一张表：n^2，n^3，n^4，n^5，要求四个数连同n在同一行打印出来。

〔6－9〕 编写由键盘输入不同N值，求

$$N! + (2N)! + (3N)!$$

的值的程序。

〔6－10〕 若从题6-4(1)中去掉45行的STOP语句时，程序执行结果与原结果是否相同？为什么？若将6-4(2)中的STOP语句去掉，情况又怎样呢？为什么？

第七章　数组和下标变量

第一节　单 下 标 变 量

一、数组与下标变量的概念

我们把相同类型的一组数据称为数组。例如，在学校中各个学生的成绩构成一个数组，在工厂中各个车间的产值也构成一个数组。但是不能把成绩与产值共同列为一个数组。

既然数组是属于同一类型的数，因此可用同一个名字来表示，每个数用不同的下标（通常按1，2，3，4，…的顺序排列）来区分。例如：A_1，A_2，A_3，A_4，A_5组成一个数组。

具有一个下标的数组称为一维数组。数组中每个数据称为数组元素。数组元素可用变量来表示，这个变量称为下标变量。

二、对下标变量的某些规定

1．数组的名称只能用英文字母A～Z中任何一个表示。有些机允许用一个字母后跟一个数字作为数组名称。

2．下标必须用圆括号括起来。

3．下标的表示方法：下标可以是数值，如A(1)，A(2)等；下标可以是简单变量，如A(X)，A(Y)；下标可以是表达式，如A(2 * X)，当X＝4时，A(2 * X)就是A(8)；下标也可以是下标变量，如A(B(1))，当B(1)＝2时，A(B(1))就是A(2)。

4. 下标如带有小数，则BASIC系统会将它以四舍五入的方式自动取整。如A(2.5)相当于A(3)，A(1.4)相当于A(1)。

5. R1机BASIC系统规定，下标的起始值是1而不是0。

下面以一个程序及其执行结果来说明上述的规定。

```
10  DIM  A(3)
20  DIM  B(3)
30  X=1.4
40  Y=2.5
50  A(1)=5
60  A(2)=6
70  A(3)=7
80  B(1)=2
90  PRINT  A(X), A(Y)
100 PRINT  A(2*X), A(B(1))
RUN
 5          7
 7          6
```

三、数组说明语句（DIM语句）

1. 格式

DIM‹数组名›（‹数值或已赋值变量表达式›）

2. DIM语句的作用

执行这个语句时，计算机根据DIM语句所说明的数组的元素个数，为数组留出存贮单元。如：

```
10  DIM  A(10)
```

计算机会留出10个单元来贮存这个数组。

如果在不采用DIM语句的情况下使用数组，这种处理方法在R 1 机BASIC系统中被认为是错误的。

3．规定

（1）在BASIC系统中规定一维数组最多只能有256个元素，而R 1 机所允许使用的数组元素个数小于这个数值。

（2）在R1机中每一个数组说明语句只能说明一个数组。如以下写法是允许的：

```
10  DIM  A(10)
20  DIM  B(20)
```

但不允许写成

```
10  DIM  A(10), B(20)
```

（3）在同一程序中，允许对数组重新说明。数组重新说明以后，计算机就把数组中的所有元素都清零。如

```
10  DIM  A(10)
20  A(3)=7
30  PRINT  A(3)
40  DIM  A(5)
50  PRINT  A(3)
RUN
     7
     0
```

四、下标变量的运用

在求一个三元素数组的平均值时，可以用下面三种方法来编程。

程序1（用简单变量编写程序）：

```
10  INPUT  A1
```

```
20  INPUT  A2
30  INPUT  A3
40  K=A1+A2+A3
50  K=K/3
60  PRINT  K
```

程序2（用下标变量而不用循环语句编写程序）：

```
5   DIM  A(3)
10  INPUT  A(1)
20  INPUT  A(2)
30  INPUT  A(3)
40  K=A(1)+A(2)+A(3)
50  K=K/3
60  PRINT  K
```

程序3（用下标变量和FOR-NEXT语句编写程序）：

```
5   DIM  A(3)
10  K=0
20  FOR  I=1  TO  3
30  INPUT  A(I)
40  K=K+A(I)
50  NEXT  I
60  K=K/3
70  PRINT  K
```

把程序3与程序1、程序2作一比较，我们可以发现：如果要处理100个元素的数组，都要编写100个输入语句，求和表达式也要写100项。程序3采取下标变量与循环语句配合使用，可使程序大大简化。例如打印元素A(1)到A(100)的值，可以把程序编写为

```
10  DIM  A(100)
20  FOR  I=1  TO  100
30  PRINT  "A(";I;")=";A(I)
40  NEXT  I
```

使用这个程序时，已经假定元素A（1）到A（100）的值被贮存在内存器之中，可以逐个取出打印。

【例1】给数组P置零值和给数组A的各元素输入数值。

程序编写如下：

```
10  INPUT  N
20  DIM  P(N)
30  DIM  A(N)
40  FOR  I=1  TO  N
50  P(I)=0
60  INPUT  A(I)
70  NEXT  I
72  FOR  I=1  TO  N
74  PRINT  A(I)
76  NEXT  I
```

在这个程序中，给数组P置零值的方法是使用赋值语句。给数组A各元素输入数值，要用INPUT语句或READ/DATA语句来编写，如用赋值语句来编写就十分麻烦了。程序中72～76行语句的编写，是为了便于检查数据（检查无误后，再把它们删掉）。本程序共用了两个循环，因为它们是互不相关的，所以允许使用同一循环变量I。

【例2】给出10个数，编写一个排序程序，将它们按由大至小的次序排列。

（1）排序方法

①把给出的10个数输入到X数组中，给定的10个数如表 7-1 所示。

表7-1

数据	69	35	55	83	75	90	85	88	65	100
变量	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	X(7)	X(8)	X(9)	X(10)

②比较换位：先将数组X(1)与X(2)比较，若X(1)＞X(2)，则原位不变；若X(2)＞X(1)时，就将X(1)与X(2)的数值交换位置。然后继续将X(1)与其它各数逐一进行比较和换位，步骤同上。例如X(1)与X(3)比较，若X(3)＞X(1)，就将X(1)与X(3)互换位置，否则X(1)与X(3)的位置保持不动。

这里要注意的是：不要把X(1)到X(10)的值看作一成不变。表7-1中数据在输入给各变量数值后，就开始进行比较换位了。当X(1)与X(4)比较时，X(4)值较大，于是需要互换位置，此时X(1)的值已改变为83。与X(6)比较换位后，X(1)值又改为90了。最后一次的比较结果：X(1)值是100。

由以上的比较换位过程可知，凡是比X(1)大的数，就作为X(1)的新值。那么，第一轮比较结束后，X(1)就是最大的数。

用同样方法进行第二轮比较。将X(2)逐次与X(3)，X(4)，…，X(10)比较换位。第二轮比较完毕后，X(2)就是仅次于X(1)的大数了。

继续进行下几轮的比较，最后一轮比较就是X(9)与X(10)进行比较换位。

上述各轮比较就是程序中外循环的内容，各次逐一比较就是程序中内循环的内容。

现在剩下的问题是如何进行互换位置。讨论之前，先打一个比喻：甲乙两人看电影时需要互换座位，最先是甲站到座位前的空处，然后乙进入甲的座位，最后在空处的甲进入原来乙的座位，这种方法称三角对换法，如图7－1所示。现在X(I)与X(J)互换也采用这种方法，第一步将X(I)值赋值给变量T，第二步将X(J)值赋给X(I)，第三步将T值赋值给X(J)。根据这个想法，写出如下赋值语句：

```
110   T = X(I)
120   X(I) = X(J)
130   X(J) = T
```

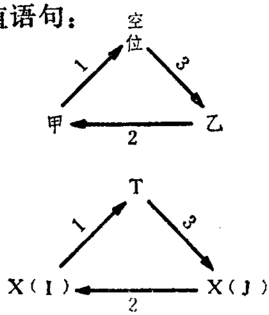


图7-1

（2）画框图

框图的次序是：

①DIM语句；

②输入数据；

③比较换位（要用到双重循环）；

④打印输出。

框图如图7－2所示。

（3）程序编写

```
10   DIM   X(10)
20   FOR   I=1   TO   10
30   INPUT   X(I)
40   NEXT   I
50   PRINT   "I", "X"
60   FOR   I=1   TO   10
70   FOR   J=I   TO   10
```

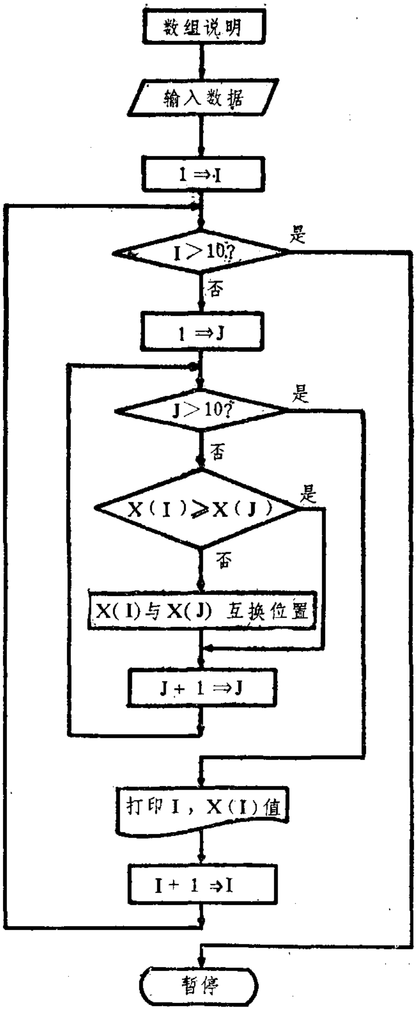

数组说明
输入数据
1 ⇒I
I＞10?
是
否
1 ⇒J
J＞10?
是
否
X(I)≥X(J)
是
否
X(I)与X(J) 互换位置
J+1 ⇒J
打印I，X(I)值
I+1 ⇒I
暂停

图7-2

```
80  IF  X(I)>=X(J)  THEN  GOTO  120
90  T=X(I)
100  X(I)=X(J)
110  X(J)=T
120  NEXT  J
130  PRINT  I, X(I)
140  NEXT  I
150  STOP
RUN
I         X
1         100
2         90
3         88
4         85
5         83
6         75
7         69
8         65
9         55
10        35
```

在这个程序中，外循环变量 I 是从 1 变化到10，而不是从 1 变化到 9，其原因是考虑到要把最后一个数打印出来。内循环变量 J 也不应从I+1开始，这是由于当 I=10时，J=11，J超出了下标值规定的范围。

这个程序有多种不同的编写方法，一种方法可以把以上程序中说明语句的X(10)改为X(11)，此时70行相应改为

```
70  FOR  J=I+1  TO  10
```

另一种方法可以用以下语句 代 替 以 上程序中的60行和70行语句：

```
60  I=1  TO  9
70  J=I+1  TO  10
```

此时需要在外循环结束之后增加一句

```
145  PRINT  10, X(10)
```

值得注意的是：由于R1机不允许使用READ/DATA语句，所以程序中输入数据采用了INPUT语句。在作另 一 次运行时，如果打入RUN及按ENTER键，计算机会把全部数据清除，此时又要重新输入数据；如果原数据不变，则只需执行60行以后的语句，此时应打入

GOTO 60 按ENTER键，

但不要打入

RUN 按ENTER键或者RUN 60 按ENTER键；

如果需要改变其中某一个变量值，则可用键盘操作方法打入不带行号的赋值语句。

【例3】利用下标变量编写一个程序，求出学习成绩为5分、4分、3分和2分的学生人数？

设下标变量A(5)，A(4)，A(3)，A(2)分别用来累计成绩为5分、4分、3分和2分的学生人数，它们的初值均应置0。学生成绩X用键盘输入，以X=－1时作为数据输入结束的标志，每当键盘输入5分成绩时，就将A(5)加1，余类推。

编写程序如下：

```
10  DIM  A(5)
20  FOR  I=2  TO  5
30  A(I)=0
```

```
40  NEXT  I
50  INPUT  X
60  IF  X=-1  THEN  GOTO  90
70  A(X)=A(X)+1
80  GOTO  50
90  FOR  I=2  TO  5
100  PRINT  "A("; I; ")="; A(I)
110  NEXT  I
RUN
```

用键盘输入数据：

5，3，2，4，5，4，5，3，4，5，-1

执行结果为

```
A(2)=1
A(3)=2
A(4)=3
A(5)=4
```

第二节 双下标变量

一、双下标变量

在实际的编程过程中，往往会碰到这样的问题，对某些性质相同的数据，我们仅用一个下标就可以把它们表示清楚，而对某些数据则必须用两个下标才能把它们表示清楚。例如表7-2所示的学生成绩管理表，它要用到几种不同变量：简单变量、单下标变量和双下标变量。

在表7-2中，编号I是简单变量。学号、总成绩和平均成绩这几栏数据，只需用Y(I)，S(I)和A(I)这三个单下

表7-2

编号 I	学号（或姓名）Y（I）	科目					总成绩 S(I)	平均成绩 A(I)
		政治 Q_1	体育 Q_2	数学 Q_3	物理 Q_4	化学 Q_5		
1	Y(1)	T(1,1)	T(1,2)	T(1,3)	T(1,4)	T(1,5)	S(1)	A(1)
2	Y(2)	T(2,1)	T(2,2)	T(2,3)	T(2,4)	T(2,5)	S(2)	A(2)
⋮		T(I,J)						

标变量就可以表示清楚了。而各科成绩包含有不同学生和不同学科，必须用两个下标才能区分。例如T(1，1)表示第一个学生的第一科成绩，T(1，2)和T(2，1)则分别表示第一个学生的第二科成绩和第二个学生的第一科成绩。因此在各科成绩栏中，数据应该用双下标变量才能清楚地把它们区别开来。表7-2用双下标变量T(I，J)表示学生各科的成绩，第一个下标I用来表示学生，第二个下标J用来表示学科。

具有两个下标的数组称为二维数组。用两个下标表示的变量称为双下标变量。双下标变量的有关规定与单下标变量的规定基本相同，这里就不再一一叙述了。

在双下标变量中的两个下标必须用括号括起来，它们之间要用逗号隔开。

如双下标说明语句

10 DIM T(10，5)

是用来说明：第一个下标的最大值是10，第二个下标的最大

值是5。这个变量所包含的元素个数的最大值是(10×5)，即50个元素，这从表7-2中容易得到理解。计算机执行这条语句时，会留出50个存贮单元供存贮数据之用。此时如果程序中的数组元素出现T(11，4)，T(5，6)时，它们均被认为是错误的。

二、双下标变量的程序举例

【例4】编写一个程序：全班学生人数为10人，共有五门课程。求每个学生的总成绩和平均成绩、全班各科平均成绩和全班总的平均成绩，并打印出来。

1．确定变量和数组说明语句

学生成绩表格如表7-2所示。表中的编号采用简单变量I；学生各科成绩属二维数组，用双下标变量T(I，J)表示；每个学生总成绩、平均成绩属单下标变量，分别用S(I)，A(I)表示；全班各科平均成绩Q(J)属于单下标变量；总的平均成绩用简单变量A表示。

下标I（I=1，2，…，10）表示第I个学生；下标J（J=1，2，…，5）表示第J门科目。

用数组说明语句说明的数组可定为S(10)，A(10)，Q(5)，T(10，5)，即告诉计算机留出75个存贮单元（也可多留出一些存贮单元）。

2．确定算式

（1）求每个学生总成绩的算式

第1个学生的总成绩为

S(1)=T(1,1)+T(1,2)+T(1,3)+T(1,4)+T(1,5)；

第2个学生的总成绩为

S(2)=T(2,1)+T(2,2)+T(2,3)+T(2,4)+T(2,5)。

写成统一的赋值式为

$$S(I)=S(I)+T(I,J),$$

其中初值为

$$S(I)=0。$$

计算每个学生的总成绩时，需要采用双重循环：

外循环变量 I， I 从 1 变化到10；

内循环变量 J， J 从 1 变化到 5。

（2）求每个学生平均成绩的算式

第 1 个学生平均成绩为

$$A(1)=S(1)/5,$$

第 2 个学生平均成绩为

$$A(2)=S(2)/5。$$

写成统一的表达式为

$$A(I)=S(I)/5,$$

此式应放在外循环体内。

（3）全班各科平均成绩的算式

$$Q(1)=(T(1,1)+T(2,1)+\cdots+T(10,1))/10,$$

$$Q(2)=(T(1,2)+T(2,2)+\cdots+T(10,2))/10。$$

写成统一的赋值式为

$$Q(J)=Q(J)+T(I,J)。$$

其中 Q(J)的初值为 0。累加后用赋值式

$$Q(J)=Q(J)/10。$$

这个过程同样需要用双重循环，不过此时内循环变量应是 I，外循环变量是 J。

（4）全班总平均成绩为

$$A=Q(1)+Q(2)+\cdots+Q(5),$$

或者写成

$$A=A+Q(J),$$

其中A的初值为0。累加后用赋值式

A＝A／5。

（5）程序的安排

数组说明语句—→输入数据（每个学生成绩）—→计算每个学生总成绩和平均成绩并打印出来—→计算全班各科平均成绩和总平均成绩。

3．画框图（见图7－3）

4．根据框图写出程序

```
10  DIM  T(10,5)
20  DIM  S(10)
30  DIM  A(10)
40  DIM  Q(5)
50  FOR  I=1  TO  10
60  FOR  J=1  TO  5
70  INPUT  T(I, J)
80  NEXT  J
90  NEXT  I
100  FOR  I=1  TO  10
110  S(I)=0
120  FOR  J=1  TO  5
130  S(I)=S(I)+T(I, J)
140  NEXT  J
150  A(I)=S(I)/5
160  PRINT  I; "␣␣"; S(I); "␣␣"; A(I)
170  NEXT  I
180  A=0
190  FOR  J=1  TO  5
```

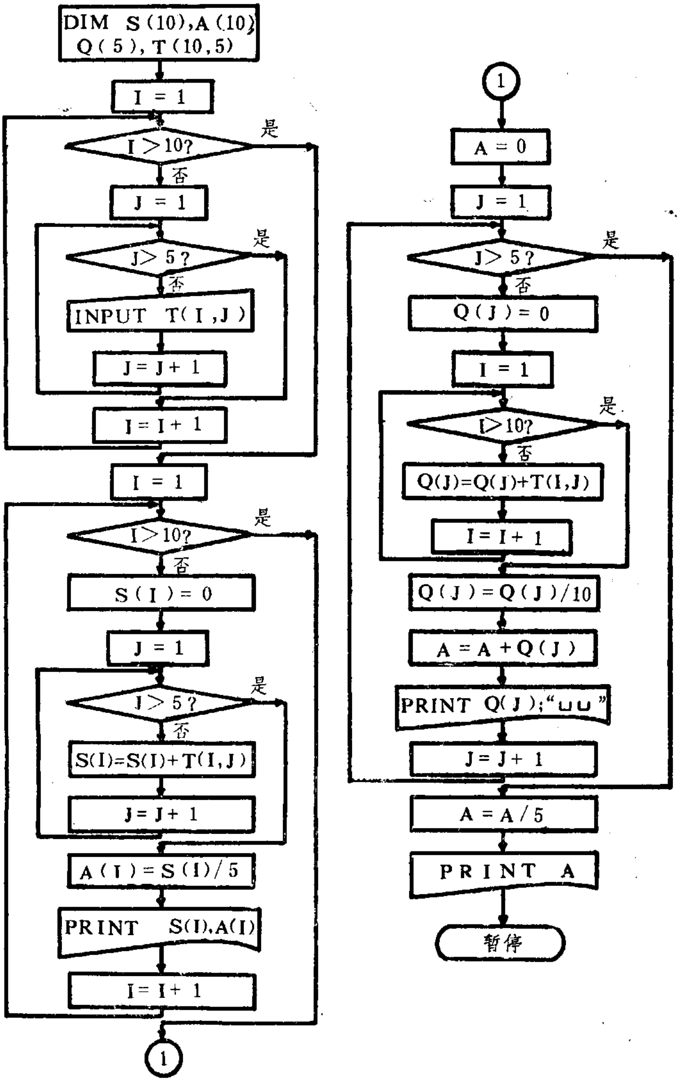

DIM S(10),A(10),
Q(5),T(10,5)
I = 1
I>10?
是
否
J = 1
J>5?
是
否
INPUT T(I,J)
J=J+1
I=I+1
I = 1
I>10?
是
否
S(I)=0
J = 1
J>5?
是
否
S(I)=S(I)+T(I,J)
J=J+1
A(I)=S(I)/5
PRINT S(I),A(I)
I=I+1
1
1
A=0
J=1
J>5?
是
否
Q(J)=0
I=1
I>10?
是
否
Q(J)=Q(J)+T(I,J)
I=I+1
Q(J)=Q(J)/10
A=A+Q(J)
PRINT Q(J);"␣␣"
J=J+1
A=A/5
PRINT A
暂停

图 7-3

```
200  Q(J)=0
210  FOR  I=1  TO  10
220  Q(J)=Q(J)+T(I, J)
230  NEXT  I
240  Q(J)=Q(J)/10
250  A=A+Q(J)
260  PRINT  Q(J); "␣␣";
270  NEXT  J
280  A=A/5
290  PRINT  A
800  STOP
```

习　　题

〔7-1〕　在以下的变量中，哪些是正确的下标变量？下标变量的维数是多少？下标是什么？（设A，B，C之值均为已知）

（1）X　　（2）X2　　（3）X(Z)

（4）Y(A, 2)　　（5）YA

（6）Y(A-B)　　（7）Y((A+B+C)*2)

（8）Y(A-B, B-C)

〔7-2〕　给出两维数组

$$A=\begin{pmatrix}5 & 2 & 0\\ 1 & 3 & 0\\ 3 & 1 & 1\end{pmatrix}$$

并已知X=2，Y=1，

（1）问A(X, Y)，A(X**Y, X-Y)，A(1

+Y * 2, 2）的值各是多少？

（2）问 A（A(2,1), A(3,2)），A（2 * A（3，X），A（X, Y）* A（Y, X））的值各是多少？

〔7-3〕 写出下面程序的执行结果：

（1）
```
10 DIM A(5)
15 ? "MAT␣A=(";
20 FOR I=1 TO 5
30 A(I)=I*2
40 ? A(I); ",";
50 NEXT I
60 ? ")"
70 STOP
```

（2）
```
10 DIM A(3,3)
20 FOR I=1 TO 3
30 FOR J=1 TO 3
40 A(I, J)=I+J
50 ? A(I, J); "␣␣";
60 NEXT J
70 ?
75 ?
80 NEXT I
```

〔7-4〕 有一无穷序列，其首项$u_1=2$，通项$u_N=N^2+1$，用单下标变量打印出此数列。要求打印格式为

$$u_1, u_2, u_3, u_4, u_5, \cdots, u_N, \cdots$$

上式中前五项均指具体值，最后一项u_N要求打印N * N+1。

〔7-5〕 用双下标变量编一程序，建立并打印一个对角线元素为1，其余元素为0的10×10的数组。

〔7-6〕 写出一个程序，把数组A改写成数组B，并将A，B两数组打印出来。已知

MAT A

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

MAT B

$$\begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}。$$

〔7-7〕 编写一个计算2×4矩阵加法的程序。

说明：矩阵加法定义如下：

$$设 A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{pmatrix}, B=\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \end{pmatrix}$$

$$则 A+B=\begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} & a_{14}+b_{14} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} & a_{24}+b_{24} \end{pmatrix}$$

第八章　字　符　串

第一节　字符串变量

在日常工作中，我们经常会遇到不少非数值的数字和符号，例如图书馆的书号TP308-54.8/4(4)，银行帐号3-098720等等，一般把这些数字符号称为字符串。具体说来，字符串就是由英文字母、数字、空格以及其他有效字符（如键符RESET作一个字符处理）组成的。

由具体的字符组成的字符串称为字符串常量。如"BEIJING"（北京）、"GOOD␣MORNING"、"1 2 3， 4 5 6 "等。字符串一般要用双引号括起来。

在编制一个需要多次引用或打印某个长字符串的程序中，如果每次都要重复书写这个长字符串，就显得十分麻烦。如果用字符串变量来表示此字符串，则可简化程序的书写工作。不过，字符串变量的实际意义仅用来存贮字符串常量。

字符串变量的表示方法是在一个英文字母后紧跟一个符号＄。如A＄，B＄等。

【例1】

```
10  DIM   A$(10)
20  DIM   B$(10)
30  A$="BEIJING"
40  B$="SHANGHAI"
50  PRINT  A$, B$
```

```
60  PRINT  B$, A$
70  STOP
RUN
BEIJING    SHANGHAI
SHANGHAI    BEIJING
```

行10和行20的语句是字符串变量的说明语句，以下我们将加以详细讨论。

使用字符串变量时，要遵循如下的规定。

1．一般在使用前，先对字符串变量的长度加以说明，说明该字符串变量最多可容纳的字符数（一个空格也算一个字符）。如上例行10和行20的说明语句，它们是用来说明字符串A$和B$的最大长度的。这里所说的字符串长度，是指字符串所包含的字符个数。

实际上，字符串变量的字符数目可以小于说明语句说明的长度，但不能超过说明语句说明的长度。R1微型机 BASIC系统规定，如字符串的实际长度超过说明的长度时，则只取说明语句中所说明的长度。例如：

```
10  DIM  A$(3)
20  A$="GOOD"
30  PRINT  A$
RUN
GOO
```

R1机的BASIC系统允许对字符串变量事先不加说明语句。此时不作为错误处理，而且在程序中出现的字符串长度几乎不受长度的限制，但这样做会浪费一些存贮单元。

2．R1机的BASIC系统规定：一个DIM语句只能说明一个字符串变量的长度，否则作为错误处理。例如：

```
10  DIM  A$(10), B$(10)
```

这种写法在R1机是不允许的，必须把它改写为两个语句。如

```
10  DIM  A$(10)
20  DIM  B$(10)
```

3. 赋给字符串变量的字符串，必须用引号把它括起来。

4. 如果使用的字符串变量事先未被赋值，则把它当作空格字符串来处理。例如：

```
10  DIM  A$(10)
20  DIM  B$(20)
30  A$ = "BEIJING"
40  PRINT  A$, B$
RUN
BEIJING
```

第二节 字符串的输入和输出

与简单变量和下标变量一样，字符串变量同样可以使用LET语句、INPUT语句和PRINT语句，用法基本相同，所以这里只作简单的说明。

1. 字符串输入

使用LET语句时，可以给一个字符串变量赋值。R1机BASIC系统规定：每一程序行只能编写一个LET语句，所赋值的字符串必须用引号括起来。

使用INPUT语句时，按规定字符串应用引号括起来。但在R1机BASIC系统中，计算机已经把INPUT语句内的字符串变量自动加上了引号，因此在打入字符串时不必另加引号，加引号反而是不正确的。此外，R1机BASIC系统规定：

一个输入语句只能输入一个字符串变量。

2．字符串输出

通过打印语句，可将一个或一组字符串常量显示出来。字符串的输出通常是利用一般形式的打印语句，要打印的字符串常量必须用引号括起来。打印输出格式有紧凑格式和分段格式两种，这与输出简单变量和下标变量时的打印格式相同。此外，字符串的输出也可以使用其它输出格式的打印语句。

下面举例说明字符串的输入和输出。

【例2】

```
10  DIM  A$(20)
20  PRINT  "WHAT  IS  YOUR  NAME"
30  INPUT  A$
40  PRINT  "MY  NAME  IS␣"; A$
RUN
WHAT  IS  YOUR  NAME
MY  NAME  IS  LI  FUNG
```

其中LI FUNG是通过键盘打进去的。

【例3】编写一个打印姓名、信箱号码和地名的程序。

```
10  DIM  A$(20)
20  DIM  B$(20)
30  DIM  D$(20)
40  PRINT  "WHAT  IS  YOUR  NAME"
              （你的名字叫什么）
50  INPUT  A$
60  PRINT  "WHAT  ARE  NO  OF  BOX
AND  CITY"
```

（信箱号码和城市名是什么）

```
70  INPUT  B$
80  INPUT  C
90  INPUT  D$
100  PRINT  A$
110  PRINT  B$; "␣"; C
120  PRINT  D$
```

在这个程序中，A$是表示姓名的字符串变量，B$是表示信箱的字符串变量，D$是表示地名的字符串变量。10行到30行是说明字符串变量的最大长度，40行到90行是计算机的提问和用户通过键盘回答，100行到120行是执行打印姓名（字符串）、信箱（字符串）、信箱号码（数值）和地名（字符串）。程序执行的结果如下：

```
RUN
WHAT IS YOUR NAME
WHAT ARE NO  OF  BOX AND CITY
CHEN HUA LIN                    （陈华林）
BOX  5012                       （5012信箱）
BEIJING                         （北京）
```

【例4】编写一个打印成绩表的程序。

```
10  DIM  A$(12)
20  DIM  C$(2)
30  PRINT  TAB(1); "COURSE"; TAB(12); "HOURS"; TAB(20); "GRADE"
```

（课程、课时、成绩）

```
40  INPUT  A$
```

```
50  INPUT  B
60  INPUT  C$
70  PRINT  TAB(1); A$; TAB(14); B;
TAB(24); C$
80  GOTO  40
```

下面是这个程序执行的假设结果（圆括号内为中文注释，并非输出结果）：

COURSE(课程)	HOURS(课时)	GRADE(成绩)
ENGLISH（英语）	3	B
SOC.(社会学)	3	A
PHYSICS（物理学）	5	C
STAT（统计学）	3	C
BUS. ADM.(企业管理)	4	B
ECON（经济学）	4	A
HISTORY（历史）	3	B
CALCULUS（微积分）	4	B

第三节　字符串运算

字符串运算包括字符串加法运算和字符串比较运算两种。

一、字符串的加法运算

我们都知道，数有加法运算，它的运算结果是一个数。而字符串加法运算的结果是将相加的字符串连接起来。

【例 5】将两个字符串相加。

```
10  DIM  A$(10)
20  DIM  B$(10)
```

```
30  A$ = "GOOD"
40  B$ = "␣MORNING"
50  C$ = A$ + B$
60  PRINT  C$
RUN
GOOD  MORNING
```

程序中50行的内容是：将A$的字符串与B$的字符串连接后“赋值”给C$变量，也可以重新“赋值”给A$或B$。

二、字符串的比较

在扩展BASIC系统中，条件语句除了可以对两个数值进行比较外，还可以对两个字符串进行比较，而且都可用六种比较符号(=，>，<，>=，<=，<>)进行比较。

字符串的比较包括两个单字符的比较和两个字符串的比较。因字符串变量必须事先赋值才能比较，所以字符串变量的比较被包含在上述两种比较之内。

1．两个单字符的比较

两个单字符比较，是通过它们所对应的ASCII代码值进行比较的。代码值大的字符为“大”。

ASCII码（参阅附录）的排列有如下规则：

数字0，1，2，…，9的代码的大小与数字的大小是一致的。即较小的数，它的代码值也较小；较大的数，它的代码值也较大。英文字母A，B，C，…，Z的代码，是按A到Z的顺序由小到大排列的。数字的代码小于字母的代码，空格的代码小于数字的代码。

由此可知，在比较两个单字符时，可按空格，0到9，A到Z的次序作前后的排列。排在较前的字符的代码，小于排在较后的字符的代码。下列的比较式是正确的：

"1"<"8","A"<"B","␣"<"0","1"<"A"。

2．两个字符串的比较

两个字符串的比较，就是从各自的第一个字符开始从左到右逐个字符进行比较。若第一个字符不相同，则以这个字符的比较结果作为整个字符串的比较结果。例如比较"ABC"与"BCD"，因"A"<"B"，所以"ABC"<"BCD"。如果第一个字符相同，则比较第二个字符；如果第二个字符也相同，则继续比较下一个字符，直到比较出两个不同字符或将全部字符比较完毕，就可以比较出两个字符串的"大小"了。例如"ABC"与"ACD"比较，比较结果为"ABC"<"ACD"；"ABCD"与"ABCE"比较，比较结果为"ABCD"<"ABCE"。

两个不同长度的字符串进行比较时，如果它们前面的字符完全相同，则可以把短字符串后面理解为补上若干空格后才进行比较，此时短字符串一定"小于"长字符串。例如"THE"<"THESE"。

对以字母组成的字符串，它们的"大小"次序与它们在字典中所排列的先后次序刚好相一致，即在字典中排列较前的字符串较"小"。

只有当它们对应位置上的字符（包括空格）完全相同时，两个字符串才能称为相等。例如："STOP"不等于"␣STOP"。

下面列举一些字符串变量用在条件语句的例句。

【例6】

```
10 IF A$<> YES THEN GOSUB 300
20 IF C$+"TIME"<R$ THEN GOTO 100
30 IF B$ THEN GOSUB 600
```

```
40  IF  A$="B"  THEN  GOTO  150
50  IF  B$=C$  THEN  PRINT  B$
```

由上例可以看出：字符串变量用在条件语句的作用与数值变量用在条件语句的作用十分相似，可以进行字符串变量与字符串比较、字符串表达式与字符串变量比较以及字符串变量与字符串变量比较。如果与空格比较不相等（30行语句的意思是如果B$不等于空格，则转向子程序600），同样可以将不等号“<>”省略。

【例7】在本章例4的成绩表里，挑选出B等成绩的各门课程，并打印出来。

```
10  DIM  A$(12)
20  DIM  C$(2)
30  INPUT  A$
40  INPUT  B
50  INPUT  C$
60  IF  C$="B"  THEN  PRINT  TAB(1);
A$; TAB(14); B; TAB(24); C$
70  IF  A$="99"  THEN  GOTO  90
80  GOTO  30
90  STOP
```

程序中60行语句是将成绩等于B的课程挑选出来打印输出。70行的"99"是终止标志，当键盘输入A$字串"99"时表示输入终止，使程序转向结束。

【例8】给出一批国家名，找出按英文字母次序排列在最前面的国家名。

本例是一个比较字符串“大小”的程序，采用的方法叫冒泡法：先任意冒出一个字符串，并假定它是“最小”者（一般

是找出第一个字符串），逐一与后面的字符串进行比较。如果前者“小于”后者，则原位不动，继续与下一个比较；如果后者“较小”，则把后者换到前面，而把原来在前面的字符串甩掉（不需互换位置）。总之，在比较过程中，始终保持冒泡的一个是“较小”的，然后再与下一个比较。比较完毕后，则第一个就是“最小”的字符串了，如按英文字母排列次序，它就是排在最前面的。

写出程序如下：

```
10  DIM   A$(10)
20  DIM   B$(10)
30  INPUT   A$
40  FOR  I=1  TO  7
50  INPUT  B$
60  IF  A$<=B$  THEN  80
70  A$=B$
80  NEXT  I
90  PRINT  A$
```

程序中60行语句的含义是：如果A$<=B$，则转向80行，继续与下一个比较；否则B$就是“较小”者，此时把B$赋值到A$中去。

假定这一批国家名是：CHINA，JAPAN，CANADA，KOREA，ENGLAND，FRANCE，AMERICA，INDIA（中国、日本、加拿大、朝鲜、英国、法国、美国、印度）。

程序执行结果为：

```
RUN
AMERICA
```

【例9】字符串运算举例。

Retro Workshop
果粉工作室

```
10  DIM  A$(5)
20  DIM  B$(5)
30  DIM  C$(15)
40  A$="NOW"
50  B$="TIME"
60  C$=A$+"IS␣THE"+B$
70  PRINT  C$
RUN
NOW  IS  THE  TIME
```

利用字符串变量以及相应的各种语句，可以对字母、数字和符号串进行加工。诸如姓名、住址、书面报告的标题等等。

第四节　子字符串

一、子字符串

整个字符串的一部分称为子字符串。子字符串的表示形式是

A$(m　TO　n)。

它的意思是：这个子字符串包括A$字符串中第m个字符到第n个字符之间的全部字符。m和n值应小于字符串A$的长度。例如：

```
10  DAM  A$(10)
20  A$="ABCDEWXYZ"
30  PRINT  A$(6  TO  9)
```

当程序执行到30行时，打印出WXYZ的字符串。在子字符串的表示形式中，有一种括弧中省略了一个数字m或n

的表示方法，如A$(m TO)表示从第m个字符到最后一个字符之间全部字符；A$(TO n)表示从第1个字符到第n个字符之间全部字符。现把上程序中添加如下语句：

```
40 PRINT A$(TO 5)
50 PRINT A$(7 TO)
```

执行到40行时，打印出字符串ABCDE；执行50行时，则打印出字符串XYZ。

如果m和n取相同的数值，就可得到字符串A$中的某一个字符。如

```
10 DIM A$(10)
20 A$="ABCDEWXYZ"
30 PRINT A$(3 TO 3), A$(6 TO 6)
RUN
C W
```

在R1机中，可将A$(3 TO 3)，A$(6 TO 6)简写为A$(3)，A$(6)。

二、子字符串的应用

【例10】删改字符串，将"GOOD MORNING"改为"GOOD BYE"。

```
10 DIM A$(15)
20 DIM B$(5)
30 A$="GOOD MORNING"
40 B$="BYE"
50 C$=A$(1 TO 5)+B$
60 PRINT C$
RUN
GOOD BYE
```

本例也可以用赋值给子字符串的办法来解答。即

```
10  DIM  A$(15)
20  A$="GOOD  MORNING"
30  PRINT  A$
40  A$(6  TO)="BYE"
50  PRINT  A$
RUN
GOOD  MORNING
GOOD  BYE
```

本程序第30行是执行打印原字符串A$，40行的内容是对A$中第6个字符以后的字符重新"赋值"。R1机BASIC系统容许重新赋值的字符个数多于原有的字符个数，但重新赋值的字符串A$的长度不得超过说明语句中所说明的长度。

【例11】编写一个程序，将英文字母A～Z按相反的顺序打印出来。

```
10  DIM  A$(26)
20  A$="ABCD…XYZ"
30  FOR  I=26  TO  1  STEP  -1
40  PRINT  A$(I)
50  NEXT  I
```

执行本程序的循环语句时，在第一次循环中，I=26，打印A$(26)，即打印出字符Z；在第二次循环中，I=25，打印A$(25)，即打印出Y；循环26次后，则从Z开始按逆次序打印到A。

第五节　LEN<字符串>函数

LEN是Length的缩写。LEN函数是用来测字符串变量的长度的。即用来计算字符串变量中包含的实际字符个数。它是一个函数，它的函数值就是字符串变量的长度。LEN函数不能作为语句命令来使用。例如：

```
10  A$="CHINA"
20  B=LEN  A$
30  PRINT  B
RUN
 5
```

在这个程序中，LEN A$的函数值"5"是CHINA的字符个数，它是一个具体的数值，所以20行的语句是把函数值赋值给简单变量B。

下面的写法是错误的：

```
10  A$="CHINA"
20  LEN  A$
```

错误之处在于20行把LEN A$作为语句来使用。

LEN函数可以出现在表达式中，因此，它可以用于带表达式的语句（如LET语句、IF-THEN语句、FOR语句和PRINT语句等）。但在LEN函数中不允许使用两个自变量，如LEN（A$，B$）是错误的；同时也不允许使用字符串相连的形式，如LEN（A$+B$）是错误的。例如：

```
10  DIM  A$(10)
20  DIM  B$(10)
30  INPUT  A$
```

```
40  INPUT  B$
50  PRINT  LEN  A$, LEN  B$
60  PRINT  LEN  A$+LEN  B$
70  PRINT  LEN  (A$+B$)
```

在这个程序中，50行和60行的写法是允许的，但70行的写法则是错误的。

【例12】字符串检索的通用程序。

给出字符串变量A$（如A$="ABCDE"），输入一批字符串，将这批字符串中凡是属于A$或A$的子字符串的字符串检索出来，并打印出"YES"和它在A$中起、止字符的位置。如：输入B$="BCD"属于A$(2 TO 4)，则检索结果为

YES 2 TO 4

打印出来。

```
10  INPUT  A$              （要求检索的字符串）
20  INPUT  B$
30  FOR  I=1  TO  (LEN  A$-LEN  B$+1)
40  C$=A$(I  TO  I+LEN  B$-1)
50  IF  B$=C$  THEN  GOTO  90
60  NEXT  I
70  PRINT  "NO"
80  GOTO  20
90  PRINT  "YES␣␣"; I; "TO"; I+LEN
B$-1
100  GOTO  20
```

本程序中，循环语句的作用是从A$中每次取出长度等于B$长度的子字符串C$，然后将B$与C$进行比较。例如：

A$="ABCDE"，B$="BCD"，先从A$中第一个字符开始，每次取三个字符作为一个字符串C$，然后用C$与B$进行比较。C$的具体取值为：第一次取出"ABC"，第二次取出"BCD"，第三次取出"CDE"。由此可以看出，循环变量的终值为

LEN A$ - LEN B$ + 1。

执行50行时，应先判断B$是否属于A$的子字符串C$。如果是的话，则转到循环体外，打印出"YES"，并打印出它在A$中起、止字符的位置。

第六节 VAL<字符串>函数和STR$<算术表达式>函数

VAL是VALUE（数值）一词的缩写。VAL<字符串>函数的作用，是将括号内的字符串转变为数值。如果括号内字符串是由数字组成，则函数值就等于组成字符串的数字之值。如：

```
VAL  "123" = 123
VAL  ("123" + "123") = 123123
VAL  ("123" + "." + "123") = 123.123
```

R1机BASIC系统规定：如果VAL函数中字符串的内容不能形成有效的数目字时，则此字符串不能用在VAL函数内，否则将会出现IE（含义是方法不对）的错误信息报告。如：

```
10  PRINT  VAL ("12" + "E" + "34")
20  PRINT  VAL  "100ABC"
RUN
```

1.2E35

IE IN 20

在10行的VAL函数括号内字符串是"12E34"，它的值代表1.2×10^{35}。而20行中的字符串"100ABC"并不代表一个有效的数目字，因此它不能用在VAL函数中。

STR＄（<算术表达式>）函数的功能正好与VAL函数的功能相反。它的功能是将括号内的算术表达式的值变换成字符串。利用它可以实现数字到字符串的转换。如：

STR＄（123）="123"

STR＄（－123）="－123"

STR＄（123－123）="0"

STR＄（123）＋STR＄（－123） ="123－123"

在以上的函数式中，第一式是将正数转换为字符串，正符号不标出来，也不留空格（TRS-80机留一空格）；第三式中括号内之值是属于数值运算，（123－123）的运算结果为零，故将字符串转换为零；最后的式子是两个函数值相加，相加的结果就是两字符串相连接，此时不能看作是数的运算。

【例13】编写一个程序：将一个十进制数（0～63）转换为八进制数。

本例给出的数字0～63，转换为八进制数后只能是一位或两位数字。设其第一位为A，第二位为B。利用STR＄函数，将A转换为A＄，将B转换为B＄，然后再将它们相加，就可得到A，B两数相连的数字（属于字符串相连，引号不打印出来）。

程序编写如下：

```
10  INPUT  X
```

```
20  A = INT ( X/8 )
30  B = X - 8 * A
40  A $ = STR $ ( A )
50  B $ = STR $ ( B )
60  X $ = A $ + B $
70  PRINT  X $
80  GOTO  10
```

【例14】编一个将数字1～100转换为数字101～200的程序。

```
10  DIM  A $ ( 3 )
20  FOR  I = 1  TO  100
30  A $ = "100"
40  A = VAL ( A $ ) + I
50  A $ = STR $ ( A )
60  SCROLL
70  PRINT  A $
80  NEXT  I
```

在这个程序中：

当I = 1时，A = 101，A $ = "101"；

当I = 2时，A = 102，A $ = "102"；

当I = 100时，A = 200，A $ = "200"。

程序的执行结果为打印出101到200之间的数字。

第七节　字符串下标变量

同一类型的字符串（如一批国家名字、课程名称等）称为字符串组。一个字符串组可用同一字符串变量名字而具有

不同的下标值来表示，即可用字符串下标变量来表示。如果程序中引用了下标变量时，一定要加上一个说明语句。字符串下标变量的格式如下：

DIM <字符变量>（P，Q）

其中：P表示最大的字符串个数，Q表示每个字符串的最大长度。例如：

```
10  DIM  A$（5，10）
```

这个说明语句定义出字符串组A$最大的字符串数为5，每个字符串的最大长度为10。它要求计算机空出 5 × 10 = 50 个内存单元。对属于这个字符串组的字符串，如果它的个数超过P值或它的长度超过了Q值，计算机会给出一个出错的信息：在显示屏的左下方出现“BS IN <行号>”，其含义是在括号内的行号中有下标不对的错误（BS是bad subscrip error的缩写）。

【例15】将一组共五个字符串输入到存贮器内。

```
10  DIM  A$（5，10）
20  FOR  I=1  TO  5
30  INPUT  A$（I）
40  NEXT  I
```

假设用键盘输入五个字符串："ABCDE","FGHIJ", "PQRST", "KLMNO", "UVXYZ", 则以上的程序代替了下面的两个语句：

```
10  READ  A$，B$，C$，D$，E$
20  DATA  "ABCDE", "FGHIJ", "PQRST",
"KLMNO", "UVXYZ"
```

由此可以看出，在处理字符串变量时，R1机不能使用

READ/DATA 语句，但可用INPUT语句来代替。这与处理数值变量所采用的方法十分相似。

现举出一个程序：

```
10  DIM  A$(10)
20  FOR  I=1  TO  5
30  INPUT  A$(I)
40  NEXT  I
```

这个程序与例15的程序看起来十分相似，只是说明语句稍有差别。正是如此，例15中的A$是属于字符串组，A$(I)表示第I个字符串。而这个程序中的A$是一个字符串变量，A$(I)指的是A$中的一个子字符串，它只含有一个字符，即第I个字符。因此，不要将上述两个程序中的A$的意义相混淆。

此外，还要注意的是：在同一个程序中，字符串下标变量与一般字符串变量的名字不能用同一英文字母去表示。

在字符串下标变量中，子字符串有如下两种表示方式：

A$($R$, M TO N)

和 A$($R$)($M$ TO N)

上两式表示字符串组A$中第$R$个字符串中的一个子字符串，这个子字符串就是从第$M$个字符到第$N$个字符之间的全部字符。例如：

```
10  DIM  A$(5, 10)
20  A$(2)="1234567890"
30  PRINT  A$(2, 3  TO  8)
40  PRINT  A$(2, 3  TO)
50  PRINT  A$(2, TO  3)
60  PRINT  A$(2, 2)
```

```
RUN
345678
34567890
123
 2
```

这个程序中，30行到60行语句可写成以下形式：

```
30  PRINT  A$(2)(3  TO  8)
40  PRINT  A$(2)(3  TO)
50  PRINT  A$(2)(TO  3)
60  PRINT  A$(2)(2)
```

习　题

〔8-1〕　判断下列式子是否成立：

(1)"CAT">"DOG"

(2)"IT　S"="ITS"

(3)LEN　"GOOD␣MORNING"=12

(4)设A$="257"，则

(a)A$+4="261"

(b)VAL　A$+4=261

〔8-2〕　写出下面程序的执行结果：

(1)10　A$="THERE␣ARE␣TWO BIRDS␣IN␣THE　TREE"

20　B$="THIS␣IS␣THREE␣DUCKS␣ON"

30　C$=A$(1　TO　10)+B$(9　TO)+"␣STREAM"

```
   40  ?   C$
   50  STOP
(2)10  A$="WHAT␣IS␣YOUR␣NAME"
   20  ?   A$
   30  B$="HIS"
   40  A$(9  TO  13)=B$
   50  ?
   60  ?   A$
   70  STOP
(3)10  A$="ALL␣THE␣COMPUTER␣
ARE␣BASED␣ON␣THE␣BINARY␣NUMBER
␣SYSTEM"
   20  ?   LEN  A$
   30  STOP
(4)10  A$="5723"
   20  A=LEN  A$
   30  B=VAL  A$
   40  P$=STR$(A)
   50  ?   A, B, P$, A$
   60  STOP
```

〔8-3〕 某运动会有中国(CHINA)、日本(JAPAN)、加拿大(CANADA)、美国(USA)四个队的二百名运动员参加。中国运动员编号为1～50，日本队编号为51～100，加拿大队编号为101～150，美国队编号为151～200。编写一程序，使当键盘输入运动员号码时，自动打印其国籍(用字符串变量存放国名)。

〔8-4〕 用字符串下标变量编写一个人事档案目录

程序。已知某单位有5人，他们分别姓王、张、李、林、刘，他们的档案分别放在1，2，3，4，5号档案袋。要求把他们的姓从键盘输入，当键盘输入其中任意一人的姓时，自动打印出对应的档案袋号。

第九章　函　　数

第一节　取整函数及其应用举例

一、取整定义及取整函数INT（X）的作用

对一个假定为X的数取整的定义是：取出不大于X值的最大整数。如果把取整的定义说成是取出比X值小的最大整数，或取出最接近X值的整数，都是不正确的。

例如：当X＝＋4时，不大于X值的整数有＋4，＋3，＋2，＋1等。而比X小的整数是＋3，＋2，＋1等。根据取整的定义，它的整数应该是＋4。

当X＝＋3.5时，不大于X值的整数有＋3，＋2，＋1等。因此取出的最大的整数是＋3。

当X＝－3.5时，不大于X值的整数有－4，－5，－6等。取出的最大的整数是－4。

取整函数INT（X）的作用，是用来产生一个不大于X值的最大整数。其中X可以是一个常数或一个变量，也可以是一个算术表达式。例如，下列的程序就是对常数取整的实例。

```
10  INPUT  X
20  Y＝INT（X）
30  PRINT  X，Y
40  GOTO  10
RUN
```

3.5	3
-3.5	-4
7.83	7
-7.83	-8

二、取整函数的应用举例

1．利用取整函数对一个数的小数位进行四舍五入的处理。例如：对3.56进行小数位的四舍五入的取整时，要改用下面的函数表达式：

INT（3.56+0.5）

即先对小数四舍五入后再取整，这时函数值就是4而不是3了。由此可看出：在INT后面的括号内加上一个常数0.5，就能起到四舍五入的作用。其一般表示式为

INT(X+0.5).

2．利用取整函数来处理一个需要保留n位小数的数，即舍去n位小数后的数。

先讨论保留小数点后两位数的情况。例如有一数为87.537，现要求保留小数点后两位数，并且要求用四舍五入的方法来处理舍去位。采用的处理方法是：先将87.537乘100得8753.7，再加上0.5，这个数变为8754.2，然后对此数取整可得8754，最后再将它÷100。写成表达式为：INT（87.537*100+0.5）/100。

由此可以得出：当利用取整函数对一个数X作保留小数点后两位的运算，且用四舍五入处理舍去位时。其函数表达式应写为

INT（X*100+0.5）/100

如果要求一个数X保留小数点后三位，且用四舍五入方法处理舍去位时，取整函数应写成

INT（X*1000+0.5）/1000

用类似的方法，可保留小数点后任意位。另外，如果不要求用四舍五入的方法来处理舍去位时，则在算式中就不用加上0.5这个常数。

8．借助取整函数，可判断一个数能否被另一个数整除。

例如，现要求判断18，17能否被3整除？很明显，18能被3整除，商为6，而INT（18/3）的函数值也是6，这两个数值恒等。17不能被3整除，其商是5.66667，而INT（17/3）的函数值为5，这两个数值不等。因此，利用取整函数来判断一个数X是否能被3整除的判断式为

INT（X/3）=X/3

如果式子两边的数恒等，则X能被3整除；如果式子两边的数不等，则X值不能被3整除。

【例1】下面是一个把能被3整除的数打印出来的程序。

```
10  INPUT  X
20  IF  INT(X/3)=X/3  THEN PRINT  X
30  GOTO  10
```

【例2】编写出分解某个数的因子的程序。

在解题前先举出一些具体的分解因式的算式，以说明题目的要求。例如：

14=1*2*7

-14=-1*2*7

4=1*2*2

1．解题思路

（1）被分解的数先由INPUT语句用键盘输入赋给变量N。用“0”作为输入数据的终止标志，当输入为0时，表

示数据全部输入完。

（2）判断N是正数还是负数？如是正数，便打印出N值＝1的字样；负数则打印出N值＝－1的字样（例如上列算式中的14＝1，－14＝－1）。然后取N的绝对值进行分解。

（3）N先被2除，每除尽一次打印一个＊2，直到不能再被2除尽为止（如上列数字中4＝1＊2＊2）。然后再把被2除不尽的商拿来被3除，每除尽一次打印一个＊3；如果被3除不尽就转去被4除，直到被$\sqrt{N}$的整数部分除为止。这一步骤可用FOR－NEXT语句来执行。

（4）判断最后的商是否为1。如果是1，则不必再打印（如上列算式中4的最后的商为1），否则打印＊（乘）最后的商（如上列算式中14的最后的商是7）。

2. 画出框图

根据上述的解题思路，拟出框图，如图9-1所示。

3. 写出程序

根据框图写出程序，框图左边的数字是程序中行号，供读者对照阅读。

```
10  INPUT  N
20  IF  N=0  THEN  GOTO  150
30  IF  SGN(N)=1  THEN  PRINT  N;"=1";
40  IF  SGN(N)=-1  THEN  PRINT  N;
"=-1";
50  N=ABS(N)
60  FOR  I=2  TO  SQR(N)
70  IF  INT  (N/I)<>(N/I)  THEN
GOTO  110
```

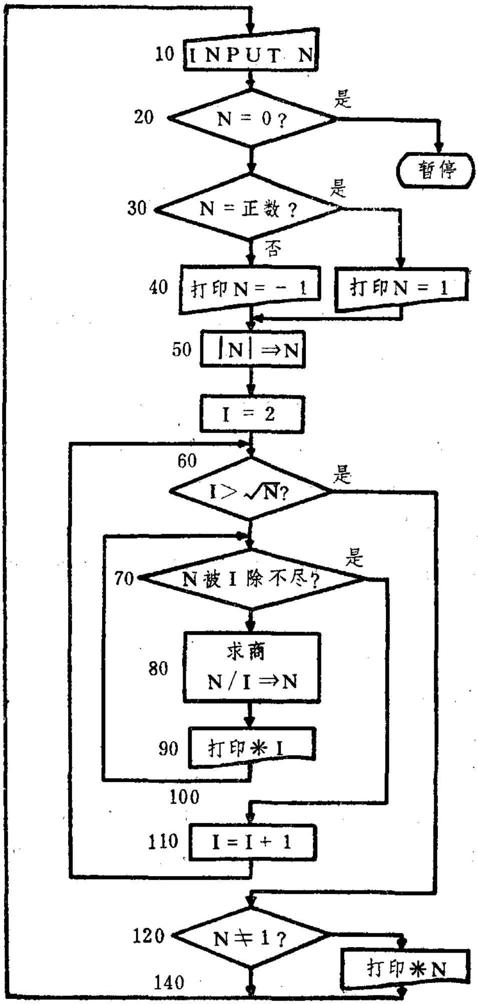

10
INPUT N
20
N = 0？
是
暂停
30
N = 正数？
是
否
40
打印N = − 1
打印N = 1
50
|N| ⇒N
I = 2
60
I＞√N?
是
70
N被I除不尽？
是
求商
80
N / I ⇒N
90
打印✳I
100
110
I = I + 1
120
N≠1？
打印✳N
140

图9-1

```
80  N=N/I
90  PRINT  "*"; I;
100  GOTO 70
110  NEXT  I
120  IF  N< >1  THEN  PRINT  "*"; N;
130  PRINT
140  GOTO  10
150  STOP
RUN
12= 1 * 2 * 2 * 3
18= 1 * 2 * 3 * 3
17= 1 * 17
-14= - 1 * 2 * 7
```

【例3】某单位财务科为了便于工资的发放，编制了一个程序，根据每个职工的实发工资，计算出到银行领款时，需要领回多少张面额分别为10元（A1）、5元（B5）、1元（B1）、5角（C5）、1角（C1）、5分（D5）、1分（D1）的钞票和工资总数，并分别打印出来。现将程序编制思路及其步骤介绍如下：

（一）分析题意，拟出算式

1. 先用INPUT语句输入每个职工的实发工资数S。用S=0作为输入终止的标志，程序转向打印结果。

2. 求每个职工应发面额为10元的钞票张数，采用下式计算：

$$W_0 = INT(S/10)$$

余数：$W = S - W_0 * 10$

此时，余数$W<10$元。

3．求面额为元、角、分的钞票张数时，采用统一算式，即各种面额（元、角、分）钞票的张数：

$$W0 = INT(W * K)$$

当W0≥5时：W5＝1，W1＝W0－5；

当W0<5时：W5＝0，W1＝W0。

式中：W5为以5为单位的钞票张数(即5元、5角和5分)；W1为以1为单位的钞票张数（即1元、1角和1分）。

余数：W＝W－W0/K

对于上述这个统一算式须作如下说明：

（1）求面额为元的张数时，取K＝1。

此时，W为上次计算的余值，W<10元。计算结果：W0是面额为元的钞票元数，W5为5元票的张数，W1为1元票的张数。余数W<1元。

（2）求面额为角的张数时，取K＝10。

计算结果：W0是面额为角的钞票角数，W5为5角票的张数，W1为1角票的张数。余数W<1角。

（3）求面额为分的张数时，取K＝100。

计算结果：W0是面额为分的钞票分数，W5为5分票的张数，W1为1分票的张数。余数W＝0。

例如，某一职工的实发工资 S＝56.73元，按照上述的算式计算结果列表如下：

表9-1

项目 \ 算式 / 计算结果	$W_0=INT(W*K)$					$W=W-W_0/K$
	K	W	W_0	W_5	W_1	
10元票的张数			5			6.73
元票的张数	1	6.73	6	1	1	0.73
角票的张数	10	0.73	7	1	2	0.03
分票的张数	100	0.03	3	0	3	0

（二）画出框图

1．子程序框图

求元、角、分票的张数时，采用了统一算式，因此可把这部分计算单独列为子程序，其框图如图9-2所示。

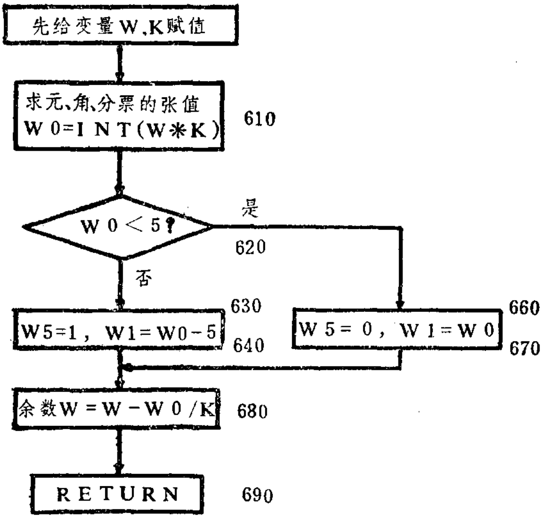


图9-2

2．整个程序的框图，如图9-3所示。

（三）根据框图写出计算程序

```
10  Q=0
15  A1=0
20  B5=0
25  B1=0
30  C5=0
35  C1=0
40  D5=0
45  D1=0
50  INPUT  S
60  Q=Q+S
```

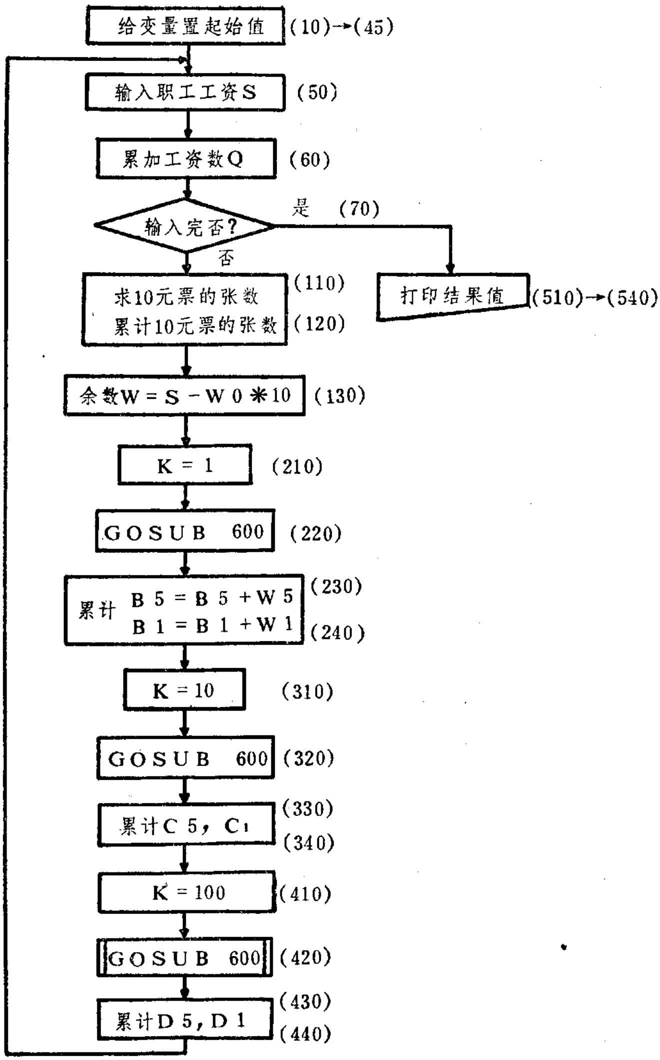

给变量置起始值
(10)→(45)
输入职工工资S
(50)
累加工资数Q
(60)
输入完否?
是
(70)
否
求10元票的张数
(110)
累计10元票的张数
(120)
打印结果值
(510)→(540)
余数W=S-W0*10
(130)
K=1
(210)
GOSUB 600
(220)
累计 B5=B5+W5
(230)
B1=B1+W1
(240)
K=10
(310)
GOSUB 600
(320)
累计C5，C1
(330)
(340)
K=100
(410)
GOSUB 600
(420)
累计D5，D1
(430)
(440)

图9-3

```
70  IF  S=0  THEN  GOTO  510
110  W0=INT  (S/10)
120  A1=A1+W0
130  W=S-W0*10
210  K=1
220  GOSUB  600
230  B5=B5+W5
240  B1=B1+W1
310  K=10
320  GOSUB  600
330  C5=C5+W5
340  C1=C1+W1
410  K=100
420  GOSUB  600
430  D5=D5+W5
440  D1=D1+W1
500  GOTO  50
510  PRINT  "Q=";  Q,  "A1=";  A1
520  PRINT  "B5=";  B5,  "B1=";  B1
530  PRINT  "C5=";  C5,  "C1=";  C1
540  PRINT  "D5=";  D5,  "D1=";  D1
600  REM  SUBROUTING（子程序）
610  W0=INT（W*K）
620  IF  W0<5  THEN  GOTO  660
630  W5=1
640  W1=W0-5
650  GOTO  680
```

```
660  W5=0
670  W1=W0
680  W=W-W0/K
690  RETURN
```

图9-3框图右侧的数字是程序的行号，以便读者互相对照，了解各语句的功能。

第二节　随机函数及其应用举例

一、随机函数RND（X）

在多数的微型机中，RND（X）表示随机函数，其中X可写任意值而不影响函数值，这里的X只是一个形式上变量而已。也有些微型机常用RND（0）表示随机函数，但这种写法在R1微型机上却被认为是错误的。R1微型机用RND表示随机函数，它把圆括弧及括号内X值都省略不写了。

利用RND函数可以产生0至1之间的随机数（不包括0和1）。例如：

```
10  FOR  I=1  TO  20
20  X=RND
30  PRINT  X
40  NEXT  I
50  STOP
RUN
```

上列程序运行一次，可产生20个0至1之间的随机数，再运行一次又产生20个不同的0至1之间的随机数，每次运行所产生的随机数几乎都不相同。如果对大量的随机数进行统计分析，则可以看出这些随机数的分布，大约有一半的随

机数分布在0至0.5之间，而另一半是分布在0.5至1之间。

利用10＊RND，可产生0至10之间的随机数（包括0而不包括10）。如果用n＊RND，则可产生0至n之间的随机数。用INT（10＊RND），则可产生0，1，2，3，…，9中任一位随机整数。同理，用INT（100＊RND）可产生0，1，2，3，…，99中任一随机整数。

在R1微型机上，还有一个与RND有关的RAND n语句，其中n值取1至65536范围内的整数(包括1而不包括65536)。用法是先打入RAND语句或命令，再引用RND函数，此时所得到的结果就不再是随机数了，而是一个对于不同的n值有一个确定值的数。例如：

```
10  RAND  1
20  PRINT  RND
30  RAND  2
40  PRINT  RND
50  RAND  3
60  PRINT  RND
70  RAND  65535
80  PRINT  RND
RUN
0.0022735596
0.0036178686
0.0045623789
0.99885559
```

经过反复试验，得到的结果都是一样的。由此可知：用RAND 1和RND可产生0.0022735596这个数，而RAND 2和RND可产生0.0036178686，RAND 3和RND产生

0.0045623789，最后一个数0.99885559是由RAND 65535和RND产生的。

二、随机函数应用举例

利用随机函数编制一段教学程序，让计算机充当一个教学机器，教小孩子做算术一位数加法的算术运算。程序的执行情况是这样的：先由计算机出题目，小孩通过键盘来回答计算结果，然后按ENTER键。如果回答对了，计算机便通过显示屏用英文批改"对了，做得好"的字样，继续出第2道题；如果小孩回答错了，计算机便告诉小孩"错了，再试一次"。

在这个程序的开头，首先应产生两个一位整数的随机数A和B，接着在显示屏上打印出算式A+B=，通过INPUT C语句，让小孩在键盘上回答计算结果。然后，用条件语句去判断答案是否正确。如果学生答案是正确的，就打印出"RIGHT, GOOD WORK"，并返回到起始部分，继续出第二道题目；如果A+B≠C，小孩子回答错了，便打印出"WRONG ANSWER, TRY AGAIN"，此时程序应转向A+B=的打印语句，而不应该返回起始部分。根据以上的程序设计思想，画出程序流程图，如图9-4所示，由此再写出程序来。

```
10  A=INT(10*RND)
20  B=INT(10*RND)
30  PRINT  A; "+"; B; "=";
40  INPUT  C
50  PRINT  C
60  IF  A+B=C  THEN  GOTO  90
70  PRINT  "WRONG  ANSWER, TRY
```

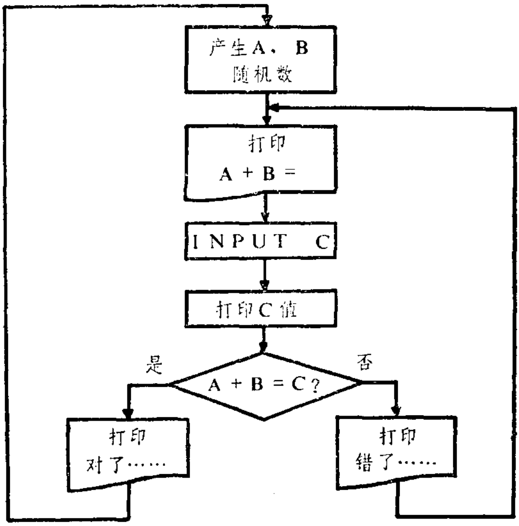


图9-4

AGAIN"

80 GOTO 30

90 PRINT "RIGHT, GOOD WORK"

100 GOTO 10

如果要求做乘法运算，则把A+B改为A*B即可。如果要计算二位整数的加法运算，则用下式产生0至99之间的随机数，即

A=INT(100*RND)

B=INT(100*RND)

第三节 打印格式函数

一、TAB(X)函数

TAB(X)函数称为列表函数，又称打印格式函数。它是PRINT语句的一种特定的输出格式，可使打印内容在由TAB(X)函数中X值所指定的那个位置上开始打印。因此，它适用于打印表格、函数和曲线。

R1微型机上的一个显示行可显示32个字符，即一行分为32格，其编号从0开始至31。每个字符打印在一个格子上。

TAB(X)函数用在打印语句的格式是

PRINT TAB(X)；〈打印内容〉

函数中的X可以是一个常数值，也可以是一个已赋值的变量或表达式。X的取值范围为 0≤X≤255。TAB(X)与打印内容之间要用分号“；”隔开。例如：

```
10  PRINT  TAB(10); "*"
20  PRINT  TAB(10); 1
30  PRINT  TAB(10); -1
40  PRINT  TAB(10.1); "*"
50  PRINT  TAB(10.5); "*"
60  PRINT  TAB(32); "*"
70  PRINT  TAB(-5); "*"
```

在上列程序中，第10行打印语句是将 * 打印在第10格位置上；第20行将 1 打印在第10格位置，R1微型机的正数不标出符号“+”；第30行也是从第10格位置上开始打印，但是在第10格上打印的是负号“-”，具体数字是从第11格开始打印的；第40行TAB函数中圆括号内含有小数，计算机会采用四舍五入的办法将它自动取整，然后把 * 打印在第10格位置上；而第50行则把 * 打印在第11格位置上；第60行圆括号内的数值超过一行所能打印的格数，计算机将自动转到下一

行第 0 格位置上打印；TAB函数的括号内的数值不容许是负值，如第70行语句的括号内出现-5，出现负值则将被认为是错误的，计算机显示出错误信息，程序也将暂停执行。

如果TAB函数中的X是已赋值的变量，那么当X值出现小数或负数时，也用上述方法处理。

如果函数的括号内是一个表达式，则打印的起始位置由表达式之值来决定。这样就使得TAB(X)函数可用来打印图形和曲线。

【例 4】

```
10  FOR  X = 0  TO  7
20  PRINT  TAB(7 - X); "*"
30  NEXT  X
```

在这个程序中，当X = 0 时，星号“*”打印在第 0 行的第 7 格的位置上。当X = 1 时，“*”打印在第一行的第 6 格位置上；……；最后第 8 个“*”打印在第 7 行的第 0 格的位置上，得到如图 9-5 所示的一根由星号“*”组成向左倾斜的直线。

【例 5】

```
10  FOR  X = 0  TO  7
20  PRINT  TAB( 7 - X); "* * * * *"
30  NEXT  X
```

运行这个程序以后，可得到如图 9-6 所示的平行四边形图形。

【例 6】

```
10  FOR  X = 0  TO  2π  STEP  π/12
20  PRINT  TAB(15 + 10 * SINX); "*"
30  NEXT  X
```

在这个程序中，X取值由 0 到2π，步长是$\pi/12$弧度（即15°）。而TAB函数中的表达式是正弦函数，因此执行这个程序的结果是打印出由 0 到2π之间，每隔15°打一个“＊”的正弦曲线。

由于SINX值含有小数，使得表达式之值亦带有小数，计算机对它取整，因此画出曲线不会很平滑。

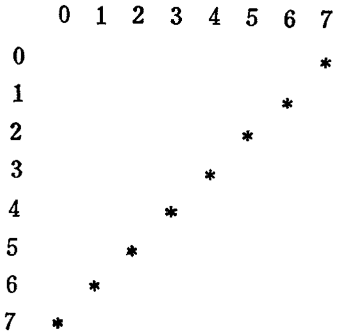


图 9－5

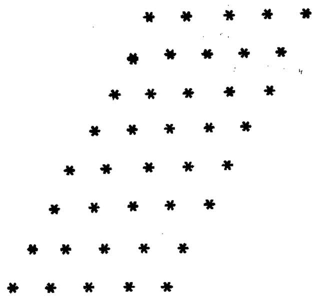


图 9－6

二、打印语句中的多个TAB(X)的函数

在一个打印语句中，容许使用多个TAB(X)函数，而每两个TAB(X)之间要用分号“；”分开。这样做使得在同一行内可以打印几项内容，这些内容分别在各个TAB（X）中X值所指定的位置上开始打印。例如：

10　PRINT　TAB（5）；A；TAB（10）；B；TAB（15）；C；TAB（20）；D

执行这个语句的结果是：在第 5 格位置上打印A值，在第10格位置上打印B值，在第15格位置上打印C值，而D值的第一个数字（如是负值则是符号）打印在第20格位置上。

这种格式对于打印表格、绘图及同时画出几个曲线是非常方便的。

但要注意：在同一打印语句中，几个TAB（X）函数中X值，只能按从左至右顺序增加，而不能减少，否则所要打印的内容就不能在同一行中打印出来。

```
10  PRINT  TAB(5); A; TAB(20); C;
TAB(10); B
```

执行这个语句的结果是在第5格位置上打印A值，然后在第20格位置上打印C值，接着是在下一行的第10格位置上打印B值，这样就显然没有达到使用者的要求。

【例7】利用TAB（X）函数打印出平方和立方表。其程序和运行结果如下：

```
10  PRINT  TAB(5); "X"; TAB(15); "X*
X"; TAB(25); "X**3"
20  FOR  X=1  TO  5
30  PRINT  TAB(5); X; TAB(15); X*X;
TAB(25); X*X*X
40  NEXT  X
RUN
X       X*X       X**3
1       1         1
2       4         8
3       9         27
4       16        64
5       25        125
```

三、TAB（X）函数和打印内容分开书写

例如：

```
10  PRINT  TAB( 5 );
20  PRINT  "*"
```

这种写法的执行结果，与打印内容在同一打印语句内的执行结果是一样的。但它的独特之处就是可以把打印内容分出来单独控制。例如：

```
10  PRINT  TAB( 5 );
20  FOR  I=1  TO  10
30  PRINT  "*";
40  NEXT  I
```

这个程序运行结果是：在第5格位置开始连续打印十个"*"号。不过要注意第30行语句末应该用分号。

【例8】编制一个程序，打印出如下的三角形图形。

```
   0  1  2  3  4  5  6  7  8  9  10 11 12 13 14
0                        *
1                     *  *  *
2                  *  *  *  *  *
3               *  *  *  *  *  *  *
4            *  *  *  *  *  *  *  *  *
5         *  *  *  *  *  *  *  *  *  *  *
6      *  *  *  *  *  *  *  *  *  *  *  *  *
7   *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
```

图9-7

这个程序可分为两段来考虑，一段是决定图形每行的"*"号的起始位置，这已在本节例1中讨论过了，另一段是决定每行打印多少个"*"号，要单独加以控制。由图形可知，每行的"*"号数是按1，3，5，…变化的，它与X行（由0算起）的关系可以用2X+1的表达式来表示，并可

以用一个循环来控制。这样就可以写出如下的程序：

```
10  FOR  X=0  TO  7
20  PRINT  TAB(7-X);
30  FOR  A=1  TO  2X+1
40  PRINT  "*";
50  NEXT  A
60  PRINT
70  PRINT
80  NEXT  X
```

程序中的外循环是用来画出图形每行的起始位置的直线，内循环是用来控制每行出现的“*”号数。

第四节 AT 函 数

在R1微型机中打印格式函数除了TAB(X)函数外，还有AT函数。

AT函数的用法与TAB(X)函数用法很相似，但也有不同之处。TAB(X)函数只能确定在一语句行中(X≤255)的某一位置上开始打印，但它究竟在哪一语句行中打印，是由程序来决定的，而不是取决于TAB函数。AT函数则可以直接指定在某行某一格位置上开始打印所需的内容。它是由打印起始位置所处的行数和列数来定位的。因此，AT函数比TAB(X)函数使用起来更灵活方便。

同样，AT函数只能用在打印语句，其格式如下：

PRINT AT <行数>,<列数>;<打印内容>

R1微型机的显示屏分为22行32列。行的编号由0到21，

按从上到下的顺序编排；列的编号由0到31，按由左到右的顺序排列。处于0行0列的格位于画面的左上角。例如：

```
1  PRINT  AT   0, 0; "*"
2  PRINT  AT   0, 31; "*"
3  PRINT  AT  21, 0; "*"
4  PRINT  AT  21, 31; "*"
```

其中，1的语句是将“*”号打印在左上角，2的语句是将“*”号打印在右上角，3和4的语句则分别将“*”号打印在左下角和右下角的位置上。

此时，语句末的标点符号不能象以前那样有换行（语句末无标志）、分段（语句末有逗号）和紧凑格式（语句末是分号）的作用。

使用AT函数有如下几个规定：

1．AT函数只能用在打印语句。

2．行与列之间要用逗号“，”分隔，而列与打印内容之间要用分号“；”分隔。

3．AT函数中行和列数，可用数值或已赋值的变量和表达式表示。打印内容可以是数字、变量（已被赋值）或字符（包括图样字符）。但要注意打印字符必须用引号括起来。

4．给出的行数或列数不能超出规定的范围，否则运行时在显示屏上会出现出错信息。如给出的列数超出允许范围，出错信息是IR。如给出的行数超出允许范围，则出现SF的出错信息。例如：

```
50  PRINT  AT  5, 32; "*"
```

执行此语句时，显示屏上出现IR IN 50，其中IR的意思是“数字出界了”。又如：

```
100  PRINT  AT  22, 5; "*"
```

执行此语句时，显示屏右下角位置上出现SF IN 100，它表示屏幕已满，也即是打印语句所给出的行数超出了允许的 0 ～21的范围。

5．一个打印语句，允许使用多个AT函数。两个AT 函数之间要用分号“；”分开。

AT函数适用于画图、写汉字、做游戏等程序。举例说明如下。

【例 9 】试编出打印一个矩形框图形的程序。

```
10  FOR  J = 0  TO  31
20  PRINT  AT  0, J; "▀"
30  PRINT  AT  21, J; "▄"
40  NEXT  J
50  FOR  I=0  TO  21
60  PRINT  AT  I, 0; "▌"
70  PRINT  AT  I, 31; "▐"
80  NEXT  I
```

这个程序的运行结果是把整个画面用一矩形框框起来。其中第20行的AT函数，行数固定在 0 行，列数由 0 到 31 变化，它是用来画出第 0 行的一根水平线。同理，第 30 行的AT函数是用来画出第21行的一根水平线。而第60 行中 AT 函数，列数固定在 0 列，行数由 0 到21变化，因此它是用来画出第 0 列的一根垂直线。而第70行的AT函数是用来画出第31列的垂直线。

由此可以得出写汉字“一横”笔划的程序：

```
10  FOR  J = <起始列> TO <终止列>
20  PRINT  AT <行数>, J; "<打印字符>"
30  NEXT  J
```

写汉字的“一竖”笔划的程序是：

```
50   FOR  I = 〈起始行〉TO〈终止行〉
60   PRINT  AT  I, 〈列数〉; "〈打印字符〉"
70   NEXT  I
```

写“一横”笔划时，只须给出该笔划所处的行和笔划的起、止列的位置共三个参数。写“一竖”笔划时，也只须给出三个参数：该笔划所处的列数和起始行及终止行数。用上述的程序，就可写出你所需要的“横和竖”的笔划。

【例10】用AT函数编制一个画斜线的程序。

利用AT函数的特点，第一个*号打印在某行某列（例如A行B列）的格子位置上，下一个*号可打印在下一行和相邻列（例A＋1和B－1）的位置上，如此连续进行下去，就可画出一条倾斜45°的斜线。下面用条件语句写出这个程序。

```
20   INPUT  C                        （*号的个数）
30   INPUT  A                        （起始点行数）
40   INPUT  B                        （起始点列数）
300  K = 0                       （K为计数用变量）
310  PRINT  AT  A, B; "*"
320  A = A + 1
330  B = B - 1
340  K = K + 1
350  IF  K<C  THEN  GOTO  310
```

程序运行时，只要打入起始点的行数A、起始点的列数B以及倾斜线点子数C，就可以在屏幕画出一条45°的直线，如图9-8所示。

如果把上列程序的第330行语句B＝B－1改为B＝B

+1，这个程序就能够画出一条为－45°的直线。

这个程序可用来写汉字"一撇"和"一捺"的笔划。写"一撇"(或"一捺")时，只要给出"一撇"(或"一捺")的起始行数A和起始列数B，以及组成"撇"(或"捺")的点的个数C，就可写出你所需要的"撇"(或"捺")的笔划。

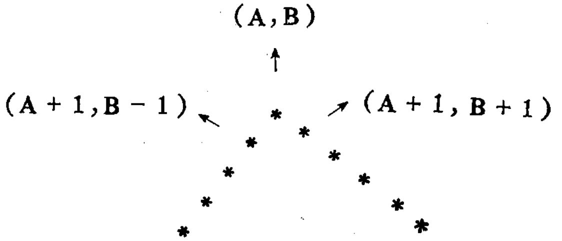


图 9-8

【例11】写任意汉字的程序。

1. 设所写汉字是由"横"、"竖"、"撇"和"捺"四种笔划组成。现在由变量N来区分上述笔划。当N=1时，写"横"的笔划；N=2时，写"竖"的笔划；N=3和N=4时，分别写"撇"和"捺"的笔划。

当N=1，写"横"的笔划时，用C变量代表"横"线所处的行，而A和B则分别代表这一横线的起始列和终止列。当N=2，写"竖"的笔划时，C变量代表"竖"笔划所处的列，A和B分别代表这一竖线的起始行和终止行。当写"撇"和"捺"的笔划时，A变量和B变量分别代表第一"撇"或第一"捺"的行和列数，C则代表组成"撇"或"捺"的点的个数。如表9-2所示。

表9-2

C	N	A	B
行　　数	1（横）	起（列）	止（列）
列　　数	2（竖）	起（行）	止（行）
撇 次 数	3（撇）	第一撇的行	第一撇的列
捺 次 数	4（捺）	第一捺的行	第一捺的行

2．写出程序的流程图，如图 9－9 所示。

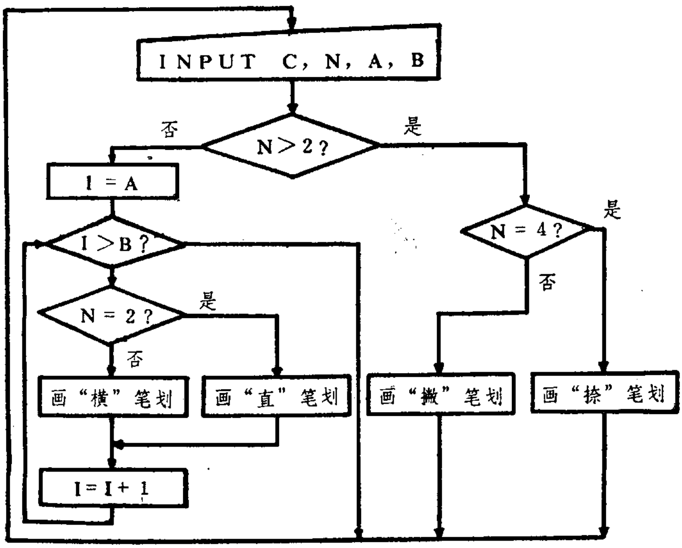


图 9－9

3．根据流程图写出如下程序：

10　INPUT　C

20　INPUT　N

```
30  INPUT  A
40  INPUT  B
100  IF  N>2  THEN  GOTO  300
110  FOR  I=A  TO  B
120  IF  N=2  THEN  PRINT  AT  I,C; "出"
130  IF  N=1  THEN  PRINT  AT  C,I; "出"
140  NEXT  I
150  GOTO  400
300  K=0
310  IF  N=3  THEN  PRINT  AT A,B;" ◢◤ "
320  IF  N=4  THEN  PRINT  AT A,B;" ◥◣ "
330  A=A+1
340  IF  N=3  THEN  B=B-1
350  IF  N=4  THEN  B=B+1
360  K=K+1
370  IF  K<C  THEN  GOTO  310
400  GOTO  10
```

根据所写汉字按笔划的顺序，写出C，N，A，B四个变量的数值，列成一个表。运行程序时，用键盘依次输入这四个变量，每输入一个变量值，都要按一次ENTER键。等到把所有变量值输入完毕后，汉字也就立即显示出来了。

表9-3就是显示“电子”两个字的变量值表。

在写入汉字时，有时要修改字的笔划，或改正打错的变量值。此时就希望有消笔划的措施，不要因为一笔一划的差错就必须重新输入数字。消笔划措施是：在程序上增添如下五个语句；但加了这些语句后，写汉字时输入的就不再是四个变量，而是五个变量了，它们的次序是C，N，A，B，M；

表9-3

C	N	A	B
4	2	6	12
6	1	5	11
12	2	6	12
9	1	5	11
12	1	5	11
8	2	4	16
17	1	8	13
4	3	5	25
5	1	19	25
1	4	9	22
10	1	18	27
2	4	16	21
23	2	11	17

这时，每画一个笔划都要键入变量值M（写笔划时M＝1）。当发现笔划有错或需要修改笔划时，只要重新键入该笔划的C，N，A，B，M值（消笔划时M＝0），就可以消去该笔划。消笔划的五个语句如下：

```
50  INPUT  M
125  IF  M=0  THEN  PRINT  AT  I, C; "␣"
135  IF  M=0  THEN  PRINT  AT  C, I; "␣"
315  IF  M=0  THEN  PRINT  AT  A, B; "␣␣"
```

```
825  IF M=0 THEN PRINT AT  A,B;"␣␣"
```

上述语句中符号“␣”代表空白键符。

习　　题

〔9－1〕　指出下列各式中X值的范围：

（1）INT（X+2）=16

（2）INT（100*X+0.5）/100=3.82

（3）INT（10*X+2）/10=4.1

（4）-INT（－10*X+0.5）/10=－2

〔9－2〕　写出下列程序运行的结果。

（1）
```
10  INPUT  A
20  ?
30  X=INT(A*100+0.5)/100
40  ?   X
50  GOTO  10
```

A变量分别取值为10.645；20.5763；13.243。

（2）
```
10  FOR X=－π  TO  π STEP  π/10
20  ?   TAB(15+15*SINX); "*"
30  NEXT  X
```

〔9－3〕　编写打印出100个在0～99的随机整数的程序。

〔9－4〕　编写一个程序：打印出如图所示格式的立方根表，要求保留到小数点后两位。

十位 \ 立方根 \ 个位	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

〔9-5〕 编写打印如下图形的程序。

```
*
**
***
****
*****
******
*******
********
*********
**********
```

```
          ***
         *****
        *******
       *********
      ***********
     *************
    ***************
   *****************
  *******************
 *********************
  *******************
   *****************
    ***************
     *************
      ***********
       *********
        *******
         *****
          ***
```

〔9-6〕 一辆卡车违反交通规则，撞死行人后畏罪逃跑了。当时有三个行人目击这一车祸的发生，但他们都只记得车牌号码的某些特征。甲和乙分别记得车牌的前两位和后两位数是分别相同的，丙是一位数学家，他说，“车牌号码肯定是四位数，且是某一整数的平方。”根据这些线索，编一程序判断肇事车辆的车牌号码。

〔9-7〕 利用随机函数产生30个100～200之间的随机整数。打印这组数据，并将它们逐个分解成由若干个素数相乘的形式。

第十章　PLOT语句、UNPLOT语句、PAUSE语句和SCROLL语句

第一节　PLOT（画图）语句

前面已经介绍了利用TAB函数和AT函数显示字符的方法。在实际应用中，我们还要经常借助计算机，画出各种图形、曲线、表格、书写汉字以及记录某一动点的轨迹等。为此，R1微型机设置了一个专用的画图语句，现讨论如下。

R1微型机的显示屏幕分为22行和32列，即共有22×32格，每格可显示一个字符。为了提高图形的分辨率，计算机又将每格分为四个象元（即四个小格，每小格称为象元，作为画图的最小单元），显示屏幕划分有44×64个象元。每个象元所在的位置，用X和Y两个坐标来定位。坐标原点定在显示屏的左下角，即把整个画面处在X、Y坐标平面的第一象限。X的取值范围由0到63，而Y的取值范围由0到43。因此，象元（0，43）位于左上角，象元（63，43）位于右上角，而象元（63，0）位于右下角。

PLOT语句就是以X，Y为坐标，以象元作为画图的最小单元的画图语句。它的格式为

```
PLOT  X, Y
```

使用这个语句时，具体规定如下：

1．X和Y的取值范围为

$0 \leqslant X \leqslant 63$

$0 \leqslant Y \leqslant 43$

2．X与Y之间用逗号分隔；

3．X，Y可以是常数、变量或表达式。如果是变量，则应事先被赋值；

4．如果X，Y是带小数的数，计算机在执行这个语句时，将自动对它取整；

5.执行这个语句的结果是：在由语句中X，Y值所确定的位置上，显示出一个象元大小的小方块，故在语句中不必另加显示字符。

对上述规定，下面举例予以说明。如：

```
10  PLOT  10, 20
```

执行这个语句的结果是在(10，20)的象元位置上显示出一个小方块。又如：

```
10  X=3
20  PLOT  X, 5
```

执行这个程序的结果是在(3，5)象元位置上显示出一个小方块。又如：

```
10  N=10
20  PLOT  N, N/3
```

在这个程序中，Y值等于3.33，计算机将对它取整，结果在(10，3)位置上显示一个小方块。

以下的几种写法是错误的：

```
10  PLOT  64, 3
```

在上语句中，X值超出了它的数值范围。又如：

```
10  PLOT  X, 5
20  STOP
```

在第10行语句中，X未被赋值。又如：

```
10  PLOT  -4, 7
```

在上语句中，X值是负值。

出现上述错误时，计算机将报告出错信息。

下面列举一些PLOT语句的应用程序。

【例 1 】画出满足函数关系Y=X的一根直线。

```
10  FOR  X=0  TO  43
20  PLOT  X, X
30  NEXT  X
40  STOP
```

执行这个程序的结果：在显示屏上显示出一条通过原点倾斜45°的直线。

【例 2 】利用PLOT画出一条正弦曲线。

```
10  FOR  X=0  TO  63
20  PLOT  X, 22+20*SIN(X*π/32)
30  NEXT  X
40  STOP
```

这个程序的运行结果：画出一条振幅为20的含有64个点子的正弦曲线

【例 3 】画出以(X, Y)为中心，中心点到每边的距离为S的正方形图形。

```
1000  REM  PROGRAM  TO  DRAW  SQUARE
      （画正方形的程序）
1010  REM  (X,Y)=CENTER  S=SIZE
      （以X，Y为中心，S为中心至每边距离）
1020  FOR  I=-S  TO  S
1030  PLOT  X+I, Y+S
1040  PLOT  X+S, Y-I
```

```
1050  PLOT  X-I, Y-S
1060  PLOT  X-S, Y+I
1070  NEXT  I
1080  RETURN
```

以上程序是画出一个正方形的通用程序，可以作为子程序使用，因此在第1080行中设置了一个返回主程序的RETURN语句。在调回此子程序时，必须事先对X，Y和S变量赋值。程序画图的顺序是按图10-1标出的1，2，3，4，5，6，7，8，…的次序进行。下面的例子是把例3的程序作为子程序来调用的举例。

【例4】画出以坐标(31，21)为中心，中心至每边的距离S为1到20的由小到大的十个正方形的图形。

```
10  X=31
20  Y=21
30  FOR  S=1  TO  20  STEP  2
40  GOSUB  1000
60  NEXT  S
70  STOP
```

（子程序参阅上例）

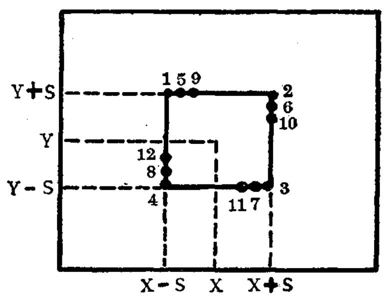


图10-1

如果把上述程序修改如下：

```
45  PAUSE  10   （停0.2秒，参阅本章第三节）
50  CLS         （消屏）
70  GOTO  30
```

这个修改后的程序的运行结果是：将逐个画出大小不断变化的正方形图形，而每个正方形在屏幕上保留的时间为0.2秒钟，然后，再画出下一个正方形，且不断循环进行下去。

第二节 UNPLOT（消象元）语句

UNPLOT语句的作用是把已经显示的象元消去，以便显示出图形的动态轨迹。其格式如下：

```
UNPLOT  X, Y
```

格式中的X和Y的含义及其规定，与PLOT语句中所述的内容相同。下面通过例子，说明UNPLOT语句的用法。

【例5】编写出画一条横线，随后又消失的程序。

```
10  FOR  I=0  TO  63
20  PLOT  I, 21
30  NEXT  I
40  FOR  I=0  TO  63
50  UNPLOT  I, 21
60  NEXT  I
70  STOP
```

这个程序包含两套循环语句，前一个循环是画出Y=21的一条横线，后一个循环又把这条横线，逐点逐点地消去。

【例6】编制一个画出一动点在水平移动的轨迹的

程序。

```
10  FOR  I=0  TO  63
20  PLOT  I, 21
30  UNPLOT  I, 21
40  NEXT  I
50  STOP
```

这个程序与例5的程序不同之处是：前者是画完一条横线后，再逐点消去的，而后者是显示一点后立即消失，程序运行结果出现一个动点在Y=21处作向右移动的轨迹。

如果希望动点的运动速度减慢，可插入一条下节所述的PAUSE语句。

第三节　PAUSE（暂停）语句

PAUSE语句与STOP语句不同之处是：STOP语句控制程序的暂停执行，一直等到使用者重新启动程序时才继续执行程序，它主要用来调试程序或作为程序的终结语句；而PAUSE语句，只使程序暂停一个很短促时间，且暂停时间由语句直接控制，它主要用来调节程序执行的时间。

PAUSE语句的格式为

PAUSE　N

其中N为整数，且$1\leqslant N<65535$。

N=1时的暂停时间为1/50秒，它是这个语句暂停的最小单位时间。如果N>32768时，则认为计算机暂停了任意长时间，此时按下键盘上任一个键，都可使程序恢复运行。

N=50时暂停时间为1秒。N=1000时暂停时间为20秒。

这个语句也可以作为键盘命令来使用。下面通过键盘命令来验证机器的暂停时间。

打入PAUSE 1000并按下ENTER键。从按下ENTER键后开始计时，直至出现OK报告为止，这段时间就是PAUSE语句的暂停时间。

下面列举一些PAUSE语句的应用例子。

【例7】数字计时器。

10 N=0

20 PAUSE 50 （暂停时间1秒）

30 PRINT AT 0，0；N （左上角显示出以秒为单位的时间）

40 N=N+1

50 GOTO 20

这个程序是一个简单的带有象征性的数字计时器，约每秒钟显示一次，从程序运行之时开始计时，单位是秒。

需要说明的是：由于计算机执行其它语句也要花时间，为了补偿这个时间差，必须把PAUSE语句中的N值调小，但它的最小单位时间是1/50秒，如果其它语句的运行时间不恰好是1/50秒的整倍数，这个程序就不能作为一个真正的秒计时器使用了。

第四节 SCROLL（滚动）语句

SCROLL语句的作用是使显示屏上的整个画面往上滚动一行，即把显示屏最顶上一行的内容消去，而把当前显示行的内容显示在最下面一行的位置上。语句的格式如下：

```
SCROLL
```

例如，有一个打印数字 1 至100的程序：

```
10  FOR  I=1  TO  100
20  PRINT  I
30  NEXT  I
40  STOP
```

执行这个程序时，由于显示屏只有22行，当循环22次以后，即进入第23次循环时，会出现一个报告信息，“SF IN 20”，其含义是显示屏满，暂停在20行。

如果要程序继续执行下去，可打入CONT命令，按ENTER键后，先清屏再显示出23～44的数字，随后又将出现显示屏满的报告信息。

要使显示屏满后，程序不中断执行而能继续显示的一种好办法，就是采用SCROLL语句。

现把上例程序改写为

```
10  FOR  I=1  TO  100
20  IF  I>22  THEN  SCROLL
30  PRINT  I
40  NEXT  I
50  STOP
```

在这个程序中，由于加插了20行的语句，当程序运行时，开始显示出 1 至22的数值，接着使整个画面往上滚动一行，从23开始以后的数字显示在最下的一行的位置上，直至循环完毕为止。

上程序又可改写为

```
10  FOR  I=1  TO  100
20  SCROLL
30  PRINT  I
```

```
40  NEXT  I
50  STOP
```

这个程序使画面在每次循环前都先滚动，然后再打印 I 值，结果在最下面一行的位置上打印出 1 至100的 数 字，每显示一个数字，整个画面就往上滚动一行。

第十一章　其它语句和函数

第一节　INKEY＄函数和游戏棒

一、INKEY＄函数

INKEY＄函数是一个特殊的字符串函数，称为读键符函数。函数值就是读出按下键的键符。例如：

```
10  K$ = INKEY$
20  PRINT  K$
```

执行这个程序的第10行语句时，如果刚好在这个时候（时间极短，只有几微秒），使用者从键盘打入了某个字符，则K＄的取值就是按下的键的键符。如果此时使用者没有按下任何键符(过早按或过迟按)，K＄取得的值就是空字符。因此只靠一个赋值语句，要读出键符值是不容易的。在实际使用中，必须把它放在循环的程序中，让计算机反复地搜索，找出被按下的键符，并且读出来。只要在上列程序中增加一条GOTO语句，就可以满足上述的要求，例如：

```
10  K$ = INKEY$
20  PRINT  K$
30  GOTO  10
```

执行这个程序时，计算机便不断地搜索键盘，只要一按下键，就立即显示出该键的键符。但这时会出现另一种情况：由于程序执行时间很短，即搜索速度很快，如果手按键的时间稍长一些，显示屏上显示出来的就不是一个键符，而

是一串相同的键符了。

解决这个矛盾的办法：可加插PAUSE语句，即加长程序的运行时间来减慢搜索速度，或者利用AT函数，使键符在固定的位置上显示出来。如：

```
10  K$ = INKEY$
20  PRINT  AT  0, 0; K$
30  GOTO  10
```

执行这个程序后，如果没有按键，则在0行0列对应的格子位置上不显示字符。而一旦打入某个键符，比如说是Y，则在该格的位置上显示出字符"Y"，并一直保持到下一次打入其它字符为止。

要显示出按下的键符，也可以用INPUT语句作为输入，通过下面的程序来实现。

```
10  INPUT  K$
20  PRINT  K$
30  GOTO  10
```

这个程序与上面的INKEY$函数的程序有什么不同呢？在这个程序中，当执行到INPUT语句时，程序会暂停执行，等待键入字符。而执行INKEY$函数的程序则不用中断程序执行，并且键入字符时，也不需要按ENTER键，计算机立即做出反应。因此INKEY$函数适合用在电子游戏的程序中。

二、INKEY$函数应用举例

【例1】写出"满天星"和"蝴蝶纷飞"的程序。

```
10  K$ = INKEY$
20  I = INT(21 * RND)
30  J = INT(31 * RND)
```

```
40  PRINT  AT  I, J; K$
60  GOTO  10
```

在这个程序中，I是0～21整数的随机值，J是0～31整数的随机值，它们分别代表打印位置的行数和列数。如按键的键符是星号“*”，不断按下此键，则显示屏上出现“满天星”。如键符是蝴蝶，则出现蝴蝶纷飞的情景。

在例1的程序中插入50行语句，可以使星星增加的速度减慢，或者使蝴蝶数的增长速度减慢。

```
50  PAUSE  10
```

如果再插入第55行的内容，使显示字符随后又消失，则出现一个蝴蝶乱飞的画面。

```
55  PRINT  AT  I, J; "␣"
```

【例2】写出利用游戏棒画图的程序。

```
10  INPUT  I
20  INPUT  J
30  K$ = INKEY$
40  IF  K$ = "C"  THEN  I = I - 1
50  IF  K$ = "V"  THEN  I = I + 1
60  IF  K$ = "B"  THEN  J = J + 1
70  IF  K$ = "N"  THEN  J = J - 1
80  PRINT  AT  I, J; "*"
90  GOTO  30
```

运行这个程序后，通过键盘输入I，J值，开始的“*”号显示在这个I，J值的格子位置上。以后每按一次C键(该键上方的键符是↑)，使“*”号上移1格；按一次V键(该键上方的键符是↓)，使“*”号下移1格；按一次B键(该键上方的键符是→)，使“*”号右移1格；按N键(该键上方

的键符是← ），使“ * ”号左移 1 格。因此，可以用C、V、B、N键来控制“ * ”的移动。而游戏棒的上下左右移动，实际上就是等效于这四个键的控制作用，从而在显示屏上画出一些有规则图形和进行游戏。程序中的“ * ”，当然也可以用其它符号来代替。

实际上，上述程序搜索键符的速度太快了，I和J值很容易超出显示屏所容许的范围，同时显示字符的移动也太快了，不便于控制掌握，此时可加插如下语句：

```
35  PAUSE  10
75  IF  I>0   THEN  I= 0
80  IF  I>21  THEN  I=21
85  IF  J<0   THEN  J= 0
90  IF  J>31  THEN  J=31
113  PRINT  AT  I, J; "␣"
116  PAUSE  5
118  PRINT  I, J; "*"
```

本程序作为 INKEY $ 函数应用的一个例子，它是通过按键去控制显示字符的移动的。实际上还可以通过按键的控制，转移到具有不同内容的程序段，或根据给定的条件选择不同的去向，把程序编得具有更丰富多采的游戏内容。

第二节　R1机的字符集、CODE函数和CHR $ 函数

一、R1机的字符集

R1 微型机的字符包括如下几类：

(1) 空格SPACE

（2）各种画图符号

（3）括号、比较符、运算符、标点符号

（4）数字 0 ～ 9

（5）字母A～Z

（6）其他键符，如：↑、↓、←、→等

（7）键入字，如：命令——NEW、RUN等

函数——SINX、COSX等

语句定义符——PRINT、PLOT等

其它——TO、STOP等

R1 微型机的字符集是指它包含的所有字符。由上看出：字符一般指的是一个符号，但有些字符是用几个英文字母表示的，例如键入字，这些字符称为有效字符。

R1微型机的字符代码也用八位二进制数表示（书写时仍用十进制），因此一个字符占用一个字节。而八位二进制共有 $2^8=256$ 种不同组合，它最多可用来表示256个字符。

实际使用时，用户通过键盘输入的及计算机最后输出的都是以上的字符，但计算机执行和处理的却是这些字符的代码（二进制）。因此，字符与代码之间就存在一个转换的问题。它们之间转换可用 CODE 函数和 CHR $ 函数来实现。前者将给定的 1 个字符转换为它的代码，而后者的功能刚好相反，将代码转换为它对应的字符。

二、CODE‹字符›函数

CODE‹字符›函数称为读代码函数，它的自变量是一个字符或有效字符，而函数值等于字符的代码。如 CODE "A" = 38。

【例 3】

```
10 INPUT A $
```

```
20  PRINT  CODE  A $
30  GOTO  10
RUN
```

打入A，按ENTER键后，显示：38

打入PRINT，按ENTER键后，显示：245

打入蝴蝶字符，按ENTER键后，显示：14

三、CHR $ 〈代码值〉函数

CHR $〈代码值〉函数称为读字符函数，它的自变量为代码值，而函数值等于代码所对应的字符。如：CHR $ 38 = A。

【例 4】

```
10  FOR  I= 0  TO  255
20  SCROLL
30  PRINT  I, CHR $ I
40  NEXT  I
```

运行这个程序以后，可在显示屏上逐一显示出 R1微型机全部字符。

有关R1微型机字符集的内容，请参阅附录。

第三节　POKE语句和PEEK函数

一、POKE〈地址，内容〉语句

POKE〈地址，内容〉语句的格式是

POKE（地址码），（内容）

其中，地址码和写入内容均可用十进制数来表示。这个函数的作用是：把语句中写入的内容存到语句中给定的随机存贮器的内存单元。由于一个内存单元只能寄存一个字节（8

位二进制数，对应的十进制数是0～255中的一个数），所以语句中写入的内容只限于一个字节。而地址码包含两个字节，即它是一个16位二进制的数，对应的十进制数为0～65535中的一个数。

R1微型机的存贮地址分配如下：ROM的地址码是0～8191之间的数（R1微型机BASIC解析程序共有8K字节）。R1微型机的基本系统只有2K RAM，它的地址码为16384～18431之间的数，如要扩展到32K RAM（需要外购一个RAM扩展板，插入到R1微型机内），扩展地址码是从18432～49151之间的数。

了解了上述地址分配情况后，使用POKE语句就不困难了。例如：

```
10  POKE  17000, 100
20  POKE  18000, 300
30  POKE  7000, 100
40  POKE  19000, 100
```

在这个程序中，第10行语句的作用是把数值100存入RAM17000的内存单元中去，第20行语句则是把数值300存入到18000的内存单元。由于一个内存单元只能存放1个字节，而300这个数超出了1个字节所能容纳的数值范围，故此语句的写法是错误的，运行结果会出现IR IN 20的报告信息。解决的办法是把300这个数用两个字节来表示，高字节为1（相当于256），低字节为44，将这两个字节分别存入到18000（存低字节）和18001（存高字节）单元。因此上列第20行语句应改写为两个语句：

```
20  POOK  18000  44
25  POOK  18001  1
```

程序中的第30行语句是把数100存入到7000的内存单元，但地址码为7000的单元是属于只读存贮器（ROM）的范围，因此，100这个数实际上是写不进去的。

第40行语句中的地址码19000属于扩展地址码，如果没有外加扩展板时，不能使用扩展地址码。

二、PEEK‹地址码›函数

这个函数的作用是读出语句中规定地址码的存贮单元内的内容。如：

```
10  PRINT  PEEK  17000
```

上述语句是由PEEK函数读出17000单元的内容，然后打印出来的。利用PEEK函数，我们可以读出ROM的内容。

【例5】

```
10  PRINT  "ADDRESS", "BYTE"
15  PAUSE  10
20  FOR  A=0  TO  8191
30  SCROLL
40  PRINT  A, PEEK  A
50  NEXT  A
```

上述程序段中，第10行语句是打印表头，ADDRESS一词是地址，BYTE一词是字节。第20行语句中0至8191是ROM的地址码。执行这个程序后，可以读出ROM中8K BASIC解析程序，并把它显示出来。

第四节 SLOW和FAST状态

R1微型机有两种工作状态：SLOW（慢）状态和FAST（快）状态。

这里的快、慢状态并不是改变计算机的运算和处理数据的速度，因为计算机的速度基本上是由硬件决定的。当计算机系统结构确定以后，它的速度也就不变了。但是用计算机处理问题所需时间，除了与计算机本身的速度有关之外，还与软件有关。因为处理同一问题，可以有不同的程序，它们的运行时间当然不会完全一样。即使在同一程序中，如果对中间过程采用不同的处理方法，也可以缩短程序的运行时间，但不改变整个程序的结果。

R1微型机慢状态是计算和显示同时进行的，遇到打印语句时立即执行。而当计算机处于快状态时，则要等到整个程序执行完毕后，才把要显示的信息同时显示出来，而在执行程序的过程中，停止显示画面，即只计算不显示，这样就缩短了程序运行时间。一般情况下，快状态和慢状态执行同一程序所需时间大约相差四倍。

当计算机第一次接通电源时，或按下RESET键和打入NEW命令后，计算机工作在慢状态。计算机在一般情况下都是在慢状态下工作的，但通过打入FAST命令可以改变程序的运行时间。

SLOW和FAST可作为语句使用而出现在程序中。如：

```
10  SLOW
20  FOR  I=1  TO  100
30  PRINT  I;
40  IF  I=51  THEN  FAST
50  NEXT  I
60  GOTO  10
```

执行这个程序时，计算机开始是在慢状态下工作的，它逐个地打印出1，2，3，4，…50的数字。当I＝51时转入快

（FAST）状态，停止了显示画面，最后才把51～100的数字一齐显示出来。

上述程序中第40行语句的比较符使用等号“=”，而不用“>=”，这是因为进入FAST状态后，只要这个状态不改变，就一直保持下去，直至I=100以后，转到第10行语句时，又进入SLOW状态。这个程序就是在两种状态下交替工作的。

第十二章　R1微型机和音乐

具有可编程序的发声功能，是R1微型机的一个出色的性能。例如，当R1微型机接通电源时，机内的喇叭会发出“噼噼”两下响声。又如，当计算机处于工作状态时，每键入一个字符，计算机都会发出不同频率的响声。当输入程序时，如果键入的语句有错，计算机给出一种蜂鸣声，用来提醒用户：现在所键入的语句有错误。利用R1机的这种发声功能，通过编程去控制声音的频率和发声的时间，就能使计算机唱出你自己编的歌曲。本节将讨论要使R1微型机发声所使用的语句或命令。

第一节　BEEP命令和NO　BEEP命令

由于R1微型机具有发声的功能，当通过键盘键入B、E、E、P四个字符并按ENTER键后，计算机就处于有声音的状态，即BEEP状态。此时按下任何键，计算机都会发出响声。

每当接通电源或按RESET键后，计算机都会自动地进入BEEP状态。所以平时使用计算机的时候，都有声音的响应。

如果觉得R1微型机有了声音太吵，要让它安静地工作，只要键入NO　BEEP等字符，然后按ENTER键，计算机就处于无声的状态了，这种状态称为NO　BEEP状态。此

时，按任何键都不会伴随有声音发生。通过上述的键盘的操作，可以很方便地控制计算机处于有声或无声的状态。

第二节　MUSIC（音乐）语句

一、音乐的常识

计算机为什么能够唱歌呢？在讨论之前我们先介绍一些音乐常识。

在一个乐器上，可以发出许多组音，基本音符有C、D、E、F、G、A、B七个。如果是C大调的话，则这七个音符分别是简谱上的1、2、3、4、5、6、7，读作do-re-mi-fa-so-la-zi。如果是D大调，则用D、E、F、G、A、B、C来表示简谱上的1、2、3、4、5、6、7。这七种自然音阶的各音之间的音程关系，有全音与半音之分。除3—4之间和7—$\dot{1}$之间是半音关系外，其余的均为全音关系，如下所示：

```
1 2 3 4 5 6 7 i̇
 V V V V V V V
 全全半全全全半
```

在某些乐器（如二胡、吉他）上，产生半音的位置与前一个音的距离比全音时少一半。在简谱上，半音的左上角位置上加注一个“#”字符，如#4等。

R1微型机BASIC系统规定：全音字符用正常使用的字符：C、D、E、F、G、A、B来表示。而半音字符则用它们的反字(白底黑字)：|C|、|D|、|E|、|F|、|G|、|A|、|B|表示。

音乐除音调以外，还有节拍之分。例如：简谱上的“5”，

表示这个音的发音持续时间为一拍；而“$\underline{5}$”则为半拍；“$\underline{\underline{5}}$”则为四分之一拍。

音乐上的音调和节拍这两个要素，用无线电学的语言来描述时，就是指声音的频率和声音的持续时间。R1微型机就是通过编程去控制声音的频率和持续时间，来奏出各种乐曲的。

二、MUSIC（音乐）语句

MUSIC（音乐）语句的格式为

MUSIC　A＄

其中，A＄是字符串变量。注意：凡是赋值给A＄的字符串，都必须要用引号括起来。如

A＄="C10D10E10C>10D>10E>10"

A＄表示一系列的音符。每个音符由两部分组成：第一部分表示音调，它用一个字符或两个字符来表示；第二部分表示发音时间的长短，用一个或两个数字来表示。对它们的表示方法有如下规定：

1．第一部分中表示音调的第一个字符是C、D、E、F、G、A、B（或它们的反字）之中的任一个，它们分别表示全音（如果是反字则表示半音）。休止符用“·”表示。

2．表示音调的第二个字符可以是“<”或“>”，也可以不要第二个字符。“<”和“>”分别代表低8音和高8音。例如：以“C”表示简谱上的“1”时，则“C<”和“C>”分别表示“$\underset{\cdot}{1}$”和“$\dot{1}$”。

3．表示声音时间长短的第二部分由1个或两个数字组成。具体地说，第二部分的数字在0到99取值。它用来表示该音调的持续时间等于多少个时间单位。若这个数是0，它

代表音调的持续时间为256个时间单位。例如："C1"表示发出"1"音的持续时间为1个时间单位，而"C10"则表示10个时间单位的时间。至于1个时间单位究竟是多少时间，它是由下述的TEMPO语句来确定的。

4. A$字符串中允许使用空格字符，但计算机在处理这个字符时，往往把它忽略了。

R1微型机所用的音符对应的字符及其频率，列在附录中。

下面的程序的写法是正确的：

```
10  A$="C10D10E10C>10D>10E>10"
20  MUSIC  A$
30  STOP
```

运行这段程序，就可听到"1-2-3-$\dot{1}$-$\dot{2}$-$\dot{3}$"的音调，而每个音调的持续时间均为10个时间单位的时间。

```
10  MUSIC  "C5D5E5C>5D>5E>5"
20  STOP
```

这个程序与上述程序的差别是速度快了一倍。

```
10  MUSIC  "C10D10E10·10C>10D>10E>10"
20  STOP
```

运行这段程序，听到的音调没有改变，但"1"音的持续时间特别长，同时中间插入了一个休止符"·10"。

下面两个程序的写法是错误的：

```
10  MUSIC  C10D10E10C>10D10E10
20  STOP
```

这个程序的错误在于音乐语句中字符串没有加引号。

```
10  MUSIC  "C100D10E5"
20  STOP
```

上一程序的错误是：第10行语句中“C100”超出了规定的音长取值范围。

第三节　TEPMO语句

TEPMO语句是用来确定MUSIC语句中时间单位的具体时间值。该语句的格式如下：

TEPMO　N

其中，N可以是一个数、变量或表达式。如果N中含有变量，则变量应先被赋值。

N值应满足：$0 \leqslant N \leqslant 255$。当$N=0$时，实际上是表示$N=256$。

当R1微型机执行到TEMPO语句时，就会把音符的发音的时间单位确定为$N \times 3.94$ms（ms是千分之一秒，即1毫秒）。例如：TEMPO　25，则1个时间单位的时间约为0.1秒。

计算机在接通电源或按RESET键后，自动地把时间单位定为0.1秒，即相当于TEMPO　25。

在一段音乐的程序里，如果适当地使用TEMPO语句，就可以调节演奏速度的快慢，使曲调变得更轻松活泼。

【例1】

```
10  A$="C1D1E1F1G1A1B1"
20  FOR  I=50  TO  250  STEP  50
30  TEMPO  I
40  MUSIC  A$
50  NEXT  I
60  STOP
```

运行这个程序时，可以明显地听出节拍越来越慢。第一遍的拍子约为0.2秒，第二遍为0.4秒，最后的一遍约为1秒。

【例 2 】下面的程序是演奏外国的名曲《苏珊娜》中的一小段。

C调　2/4　　SUSANNA

1 2 | 3 5 5·6 | 5 3 1·2 |

C2 D2　E4 G4　G6 A2　G4 E4　C6 D2

3 3 2 1 | 2 0 1 2 |

E4 E4　D4 C4　D8·4 C2 D2

```
10  REM  SUSANNA
20  A$ = "C2D2E4G4G6A2G4E4C6D2E4E4D4C4
D8·4"
30  FOR  I=1  TO  3
40  TEMPO  45/I
50  MUSIC  A$
60  MUSIC  A$
70  NEXT  I
```

本程序第40行的语句用来改变演奏的速度，第50行和60行语句的作用是使用同一速度唱两次，然后再循环三遍。

第四节　SOUND语句

SOUND（声音）语句常用在音乐程序或游戏程序里，产生一种特殊的音响效果。例如，在程序中当出现某种特殊情况时，就发出一种你所约定的声音。

这个语句的格式如下：

SOUND M，N

其中，M和N可以是数或变量。它们如果是变量，应事先被赋值。M和N值都应是整数，它们的取值范围是

$0 \leqslant M < 256$

$0 \leqslant N < 65536$

如果M＝0时，计算机在运行程序时，把它作为M＝256来处理。如N＝0，则作为N＝65536处理。

当M、N值确定后，SOUND语句产生一个以频率为32500/M赫，持续时间为N/65ms的声音。M值很大时，SOUND语句产生低音的音响效果。

第十三章　实验、操作指导

实验一　特殊功能键和键盘命令的操作

一、实验目的

1．掌握按键的操作方法。

2．弄清特殊功能键和键盘命令的功能。

3．分清特殊功能键和键盘命令的不同点。

二、实验内容

（一）按键操作

当显示屏左下角出现闪动光标，即计算机处于命令状态时，按键才起作用。按下的键的键符，均在显示屏的最下面一行位置上显示出来。

1．逐个按下键盘上第二排的键，观察显示结果。

2．先按下SHIFT键（按住不放），再逐个按一下（按下后即放）第二排的键，然后释放SHIFT键，观察显示结果。这种操作方法，以下用同时按SHIFT键和其它某个键的说法来表述。

3．显示第三个键符的操作方法。

（1）同时按SHIFT键和ENTER键，使计算机进入作图（GRAPHICS）状态。注意观察显示屏下方闪动的方块光标变为白底黑字的闪动🄶光标。

（2）逐个按下第二排的键，观察显示结果。

（3）然后同时按SHIFT键和第二排的键，观察显示

结果。

（4）按ENTER键，退出GRAPHICS状态。此时恢复正常的闪动的方块光标。

按第三排的键，重复上述操作内容。

（二）特殊功能键和键盘命令的操作

程序：

```
10  X=5
20  PRINT  X
30  Y=10
40  PRINT  Y
```

按如下步骤操作：

1．逐个打入N、E、W，按ENTER键。

2．逐行打入程序行，应用LINE NO特殊功能键编行号。

3．逐个打入R、U、N，按ENTER键。注意：先清屏再显示，RUN字符显示在左下角。

请读者想一想：按下特殊功能键后，它的字符是否被显示？是否要按ENTER键？是否有清屏功能？

4．再打入RUN，按ENTER键，观察结果是否一样。这个步骤可用来说明RUN命令不清除存入到内存贮器的程序。

5．打入LIST，按ENTER键，列出程序清单，观察程序是否一样？再一次说明RUN命令不清内存中程序。

6．打入RUN 30后，按ENTER键，不显示X值。原因是什么？读者自己考虑。

7．打入RUN 40后，按ENTER键，出现错误信息。原因是什么？读者自己考虑。

8．打入RUN后，按ENTER键。观察结果还是一样，说

明RUN命令不清内存中程序。

9. 打入CLS后，按ENTER键，其作用是清除屏幕画面。

10. 打入LIST 20后，按ENTER键。打印出以20行开始以后程序清单（有些机只列出20行内容）。

11. 打入RUN，按ENTER键后，马上按BREAK键（动作要迅速），使程序中断在20行或30行，再打入CONT命令后继续执行程序。请读者想一想，此时是从哪一行开始继续执行的？然后反复操作几次。

12. 打入20 按ENTER键，其功能是删去20行。

13. 打入20 STOP后，按ENTER键，更改20行语句。

打入RUN后，按ENTER键，此时无显示。原因是程序暂停在20行。

打入CONT后，按ENTER键，继续从30行开始执行程序，显示出Y值。

14. 比较一下特殊功能键和键盘命令的不同之处。

实验二 源程序的删改

一、实验目的

熟练掌握源程序的删改操作。

二、实验内容

设已编好程序如下：

```
10  X=3*5
20  Y=6*7
30  PRINT  "X"=X, "Y="Y
50  PRINT  "Z="; (X+Y)/2
```

1．用键盘输入上列程序，如按错键或漏按键，试删改之。

2．第30行语句有句法错误，要改为

30　PRINT　"X="; X, "Y="; Y

删改字符操作步骤：（1）按←或→键，将光标移至待删改字符的右边；（2）按DELETE键，删去光标左边的字符；（3）打入字符；（4）按ENTER键。

3．修改程序

（1）删去30行语句。方法是打入30后按ENTER键。

（2）增加第40行的内容：打入40　Z=(X+Y)/2，按ENTER键。

（3）修改第50行语句的内容。改为

50　PRINT　"Z="; Z

步骤：先按↓键使光标下移一行，再按EDIT键及连按→键，使光标移至最后，然后按DELETE键，每按一次，删去光标左面一个字符，连续按七次删去(X+Y)/2，打入字符Z，最后按ENTER键。

实验三　键盘运算

一、实验目的

1．熟悉用键盘进行直接运算。

2．熟悉键盘运算在调程序时的应用。

二、实验内容

（一）直接运算

1．计算 $\sqrt{3^2+4^2}$ 和 $\sqrt{5.5^3+8.8^3}$ 的值。

键盘输入的步骤：

PRINT　SQR(3*3+4*4)　按ENTER键

A=5.5**3　按ENTER键

B=8.8**3　按ENTER键

PRINT　SQR(A+B)　按ENTER键

2. 求COS30°、TAN40°的值。

键盘输入步骤：

PRINT　COS(30*π/180)　按ENTER键

PRINT　TAN(40*π/180)　按ENTER键

3. 求$\log_{10}30\left(=\frac{\ln 30}{\ln 10}\right)$的值。

打入

PRINT　LOG30/LOG10　按ENTER键

（二）在调试程序时的应用

输入程序：

```
10　A=7*COS(π/3)
20　B=8.5*9.6
30　PRINT　SQR(A/B)
```

打入RUN　按ENTER键后进行下面的操作。

操作内容：

1. 用键盘如何检查A=？，B=？

PRINT　A　按ENTER键

PRINT　B　按ENTER键

2. 试把A、B的数值变为：A=6，B=8*9，但不许改动程序，观察结果值=？

A=6　　　按ENTER键

B=8*9　　按ENTER键

GOTO　30　按ENTER键

3. 重新运行程序，用键盘检查A、B值，观察A、B值是原来数值还是新值。

```
RUN          按ENTER键
PRINT  A     按ENTER键
PRINT  B     按ENTER键
```

实验四　打 印 语 句

一、实验目的

1. 熟悉一般格式的打印语句，掌握两种打印输出格式及打印项之间标点符号的用法。

2. 学会编程时使用打印语句。

二、实验内容

（一）上机操作下列程序，并将显示屏画面填在以下的框内。观察和分析打印输出结果。

```
1.  10  PRINT  1, 2
    20  PRINT  10
2.  10  PRINT  1; 2
    20  PRINT  10
```

1.　　　　2.

```
3.  10  PRINT  1; 2;
```

```
    20 GOTO 10
4.  10 PRINT "GOOD"; "MORNING"
    20 PRINT "GOOD"; "MORNING"
    30 PRINT "GOOD"; "␣"; "MORNING"
    40 PRINT "GOOD  MORNING"
```

3.

4.

```
5.  10 PRINT X=5
    20 PRINT "X" X
```

第20行语句原意要求打印 X=5，试改正程序后运行（先打入20行内容，显示器出现如下左图的画面，再进行修改）。

5.

```
 10▩PRINT  X=5


20  PRINT  "X"E X
```

5.

（二）用打印语句编程

1．编写显示下列表格的程序（符号␣表示空格），并上机操作。

	A	B	C	D
1				
2				
3				
4				

2. 编写打印出X = △△　Y = △△的程序（符号△△为变量值，由使用者给定。Y与X值间空一格）。

3. 编写打印出THIS IS A MODE—R1 MICROCOMPUTER 的程序，要求用两个打印语句打印。

（三）自编程序或按例题、习题中内容进行练习。

实验五　INPUT语句

一、实验目的

1. 熟悉当程序运行到INPUT语句时，如何用键盘输入数据的操作方法。

2. R1机没有READ/DATA 语句。初步了解用 INPUT 语句代替READ/DATA语句的方法。

二、实验内容

本实验中所用程序应用语句较多。有些语句是未学过的，它们将在后续章节陆续进行讨论。这里是通过实验，使读者对 INPUT 语句有一定的了解，而不是去研究各个语句在程序上的作用。

1. 商业上售货计价程序

执行这个程序的结果如表所示（中文字不打印）。售货员将顾客购买货物的数量、单价，通过键盘输入（表中××），最后键入0，表示输入数据完毕。计算机便自动进行计算，并将合计金额打印出来。顾客交来金额也通过键盘打入，计算机算好

数　　量	单位（元）
×　×	××.××
×　×	××.××
合计金额	△△.△△
交来金额	××.××
应找回金额	△△.△△

应找回的金额后并打印出来。

```
10  T=0
20  INPUT  A       (数量)
30  IF  A=0  THEN  GOTO  90
40  INPUT  B       (单价)
50  PRINT  A; "␣␣␣"; B; "␣␣␣"
60  C=A*B
70  T=T+C          (合计金额)
80  GOTO  20
90  PRINT  "␣␣␣"; T
100  INPUT  D      (交来金额)
110  PRINT  "␣␣␣"; D
120  E=D-T         (计算应找回金额)
130  PRINT  "␣␣␣␣"; E
140  GOTO  10  (转到10行)
```

程序行30行的含意是：如果A=0（表示输入数据完毕），就转到90，打印出合计金额，否则继续执行下一行（40行）。

运行程序的操作步骤如下：

（1）打入RUN后，按ENTER键。

（2）显示屏右下角出现闪动光标，表示程序运行到INPUT语句，等待输入数据。

（3）键入A数据，按ENTER键。

（4）键入B数据，按ENTER键。

（5）输入A、B数据后，键入0，按ENTER键。

（6）键入交来金额后，按ENTER键。

例：1↙2↙3↙4↙0↙15（↙表示ENTER，以下

同）

↑　↑　↑　↑　终　↑

A　B　A　B　止　D

2．求平均值的程序（设求三个数的平均值）。

```
10  INPUT  A1
20  INPUT  A2
30  INPUT  A3
40  PRINT  A1;"␣␣"; A2;"␣␣";A3; "␣␣"
50  K=A1+A2+A3
60  V=K/3
70  PRINT  K, V
```

设A1=1，A2=2，A3=3时，运行程序的操作如下：

1↙2↙3↙

3．用INPUT语句输入10个数据到内存贮器的程序。

```
10  DIM  A(10)
20  FOR  I=1  TO  10
30  INPUT  A(I)
40  PRINT  A(I); "␣␣";
50  NEXT  I
```

如果需要将存入内存贮器的数据取出来参加运算的话，可增加下列语句。

```
60  K= 0
70  FOR  I=1  TO  10
80  K=K+A(I)
90  NEXT  I
100  V=K/10
110  PRINT
```

120 PRINT　K，V

操作方法：

（1）打入RUN并按↙（符号↙表示ENTER键）。

（2）打入如下十个数据：

1↙2↙3↙4↙5↙6↙7↙8↙9↙10↙

（3）再次执行运算程序时，打入GOTO　60↙，不能打入RUN↙或打入RUN　60↙，因为它们会把全部变量清除，必须重新输入数据。

实验六　条件语句、循环语句

一、实验目的

初步学会条件语句和循环语句的应用。

二、实验内容

1.使用条件语句打印（显示）出0到12之间的偶数。程序如下：

```
10  E=0
20  PRINT  E
30  E=E+2
40  IF  E<=12  THEN  GOTO  20
```

RUN

```
     0
     2
     4
     6
     8
    10
    12
    OK  IN  40
```

练习：

（1）试打印（显示）出0到20之间的偶数。

（2）试打印（显示）出0到20之间的奇数。

（3）改用循环语句编写程序。

```
10 FOR E=0 TO 12 STEP 2
20 PRINT E
30 NEXT E
```

2.按照下列程序，试先考虑可得出什么结果，然后上机操作。

```
10 FOR C=1 TO 11
20 FOR M=1 TO 16
30 PRINT "▀ ▄";
40 NEXT M
50 FOR N=1 TO 32
60 PRINT "▚";
70 NEXT N
80 NEXT C
```

注意：本例使用循环语句时，FOR和NEXT语句必须成对出现。

3.运行下面程序并对结果值进行分析。

```
100  FOR  N=10  TO  20
110  PRINT  N;
120  FOR  J=30  TO  40
130  PRINT  J;
140  NEXT  N
150  NEXT  J
RUN  （按ENTER键）
```

```
10 30 11 30 12 30 13 30 14 30 15 30 16 30 17 30
18 30 19 30 20 30 31 32 33 34 35 36 37 38 39 40
OK  IN  150
```

这段程序之所以会产生这样的打印结果，是因为交叉使用了两个循环语句。在正常地使用循环语句时，一般是不容许交叉的。有时，为了得到特殊的效果，也可以别具匠心地使用交叉的循环语句，但必须小心为之。

实验七　下 标 变 量

一、实验目的

学会使用下标变量。

二、实验内容

1．利用下标变量将数据存入内存贮器。

（1）程序1（用简单变量输入数据）

```
10  FOR  I=1  TO  3
```

```
20  INPUT  X      （输入数据11，22，33）
30  NEXT
```

RUN （按ENTER键）

打入11 （按ENTER键）

打入22 （按ENTER键）

打入33 （按ENTER键）

这时如果利用键盘命令打印刚才输入的参数，例如：

PRINT X

按ENTER键后，只显示33。这种情况说明上列程序中因为只用了简单变量，所以不能将三个数据存入内存贮器。

（2）程序2（用下标变量输入数据）

```
10  DIM  X(3)
20  FOR  I=1  TO  3
30  INPUT  X(I)
40  NEXT  I
```

RUN （按ENTER键）

打入11 （按ENTER键）

打入22 （按ENTER键）

打入33 （按ENTER键）

（3）如果要利用下标变量将内存贮器中的数据打印出来，程序2增加下列语句：

```
50  FOR  I=1  TO  3
60  PRINT  "X("; I; ")="; X(I)
70  NEXT  I
```

GOTO 50 （按ENTER键）〔此处不能用RUN 50（ENTER）〕

程序运行结果：

```
X(1)=11
X(2)=22
X(3)=33

OK IN 70
```

2．编制一个统计学生成绩的程序，要求分出获得100分、90～99分、80～89分、……、10～19分的学生各有多少人。

程序：

```
10  DIM  S(10)
20  FOR  I=1  TO  10
30  S(I)=0
40  NEXT  I
50  INPUT  Q
60  PRINT  Q; "          ";
65  IF  Q=-1  THEN  GOTO  100
70  R=INT(Q/10)
80  S(R)=S(R)+1
90  GOTO  50
100  PRINT
110  PRINT
120  FOR  I=1  TO  10
130  IF  I=10  THEN  GOTO  160
140  PRINT  10*I; "——"; I*10+9, S(I)
150  GOTO  170
160  PRINT  "          "; I*10, S(I)
170  NEXT  I
```

本程序是利用数组S(I)来记录各类得分的学生数。其中

用S(1)记录10分到19分之间的学生数；S(2)记录20分到29分的学生数；……；S(10)记得100分的学生数。

程序中第70行语句的内容是：将成绩Q除10后取其整数。例如，成绩Q＝86时，取整函数INT(86/10)的函数值为8，赋给变量R，记入S(R)＝S(R)＋1。

执行程序的操作过程如下：

RUN （按ENTER键）

逐个地输入学生成绩，例如：90，59，100，60，89，－1。

注意：每输入一个成绩后，均应按ENTER键。最后输入－1是表示终止数据的输入，按ENTER键后显示出：

```
10——19        0
20——29        0
30——39        0
40——49        0
50——59        1
60——69        1
70——79        0
80——89        1
90——99        1
  100         1

OK  IN  170
```

3．利用双下标变量，将下表所示的三行四列数据，输入到内存贮器。

1	2	3	4
5	6	7	8
9	10	11	12

程序：

```
10  DIM  T(3,4)
20  FOR  I=1  TO  3
30  FOR  J=1  TO  4
40  INPUT  T(I, J)
45  PRINT  T(I, J); "␣␣␣";
50  NEXT  J
55  PRINT
60  NEXT  I
RUN  (按ENTER键)
```

逐个打入数据：1，2，3，4，5，6，7，8，9，10，11，12。

为了便于观察运行结果，在程序中加插了45和55两行语句。

实验八　函数、随机函数

一、实验目的

初步学会使用函数和随机函数的方法。

R1微型机键盘上有下列几种函数：

ABS　绝对值　　　　SIN　三角函数正弦

COS　三角函数余弦　　TAN　三角函数正切

ASN	反三角函数正弦	ACS	反三角函数余弦
ATN	反三角函数正切	LOG	自然对数
EXP	指数函数	SQR	开平方
RND	随机函数	SGN	符号函数

二、实验内容

1．观察ABS函数值

```
10  PRINT  ABS  5
20  PRINT  ABS-5
RUN  (按ENTER键)
5
5
```

2．观察SGN 函数值：正数取1；负数取－1。如果数是0，则取0。

```
10  PRINT  SGN5
20  PRINT  SGN-5
30  PRINT  SGN  0
RUN  (按ENTER键)
1
-1
0
```

3．随机函数RND用来产生0～1之间的随机数（不包括0和1）。

（1）产生20个0～1之间的随机数的程序：

```
10  FOR  I=1  TO  20
20  X=RND
30  PRINT  X
40  NEXT  I
```

RUN （按ENTER键）

（显示出20个不同的0～1之间的随机数）

再运行一次，观察显示结果。

（2）将上面程序第20行语句改为

```
20  X=INT(10*RND)
```

观察运行结果。

（3）将上面程序的第20行语句再改为

```
20  X=INT(100*RND)
```

观察运行结果。

4. 观察INT取整函数值

```
10  INPUT  X
20  A=INT  X
30  B=INT(X+0.5)
40  C=INT(X*10+0.5)/10
50  D=INT(X*100+0.5)/100
60  PRINT  X
70  PRINT  A
80  PRINT  B
90  PRINT  C
100  PRINT  D
RUN （按ENTER键）
```

打入9.876 （按ENTER键），观察运行结果。

打入-9.876 （按ENTER键），观察运行结果。

分析打印输出结果。

实验九 画图、写汉字

一、实验目的

1．熟悉TAB(X)函数及AT函数在打印语句中的应用。

2．熟悉PLOT语句的应用。

二、实验内容

1．打印出下面表格。

先思考一下应该选用哪个语句和函数，不要以为打印表格一定要用TAB(X)函数。

I	A	B	C
1	12	23	34
2	45	56	67
3	78	89	90

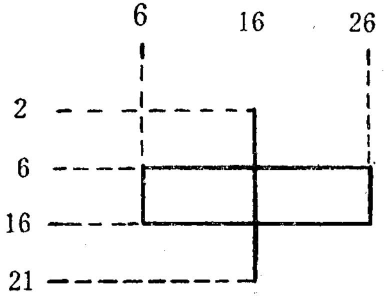
参考程序如下：

```
10  FOR  Y=2  TO  42  STEP  10
20  FOR  X=0  TO  60
30  PLOT  X, Y
40  NEXT  X
50  NEXT  Y
60  FOR  X=0  TO  60  STEP  15
70  FOR  Y=2  TO  42
80  PLOT  X, Y
90  NEXT  Y
100  NEXT  X
```

```
110 PRINT AT 3,4; "I"; AT 3, 11; "A";
AT 3, 18; "B"; AT 3, 25; "C"
120 DIM T(3, 3)
130 FOR I=1 TO 3
140 PRINT AT 5*I+3, 4; I
150 FOR J=1 TO 3
160 INPUT T(I, J)
170 PRINT AT 5*I+3, 7*J+4; T(I, J)
180 NEXT J
190 NEXT I
```

运行程序时，用键盘输入表中所列数字（可以自拟）。注意：每键入一个数字，应按ENTER键。

2. 编制一个写汉字“中”的程序。



程序1：

```
10 FOR I=6 TO 16
20 PRINT AT I, 6; "*"; AT I, 26; "*"
30 NEXT I
40 FOR I=2 TO 21
50 PRINT AT I, 16; "*"
```

```
60  NEXT  I
70  FOR  J=6  TO  26
80  PRINT  AT  6, J; "*"; AT  16, J; "*"
90  NEXT  J
```

如把“中”字中间的竖线分为两段，可以简化程序，省去一个循环。如程序 2 所示。

程序 2：

```
10  FOR  I=6  TO  16
20  PRINT  AT  I, 6; "*"; AT  I, 26; "*";
AT  I-4, 16; "*"; AT  I+5, 16; "*"
30  NEXT  I
40  FOR  J=6  TO  26
50  PRINT  AT  6, J; "*"; AT  16, J; "*"
```

3. 写任意汉字的程序。

```
10  INPUT  C
20  INPUT  N
30  INPUT  A
40  INPUT  B
50  INPUT  M
100  IF  N>2  THEN  GOTO  300
110  FOR  I=A  TO  B
120  IF  N=2  THEN  PRINT  AT  I, C; "出"
125  IF  M=0  THEN  PRINT  AT  I, C; "␣"
130  IF  N=1  THEN  PRINT  AT  C, I; "出"
135  IF  M=0  THEN  PRINT  AT  C, I; "␣"
140  NEXT  I
150  GOTO  400
```

```
300  K=0
310  IF  N=3  THEN  PRINT  AT  A, B; "▞"
315  IF  M=0  THEN  PRINT  AT  A, B; "␣␣"
320  IF  N=4  THEN  PRINT  AT  A, B; "▚"
325  IF  M=0  THEN  PRINT  AT  A, B; "␣␣"
330  A=A+1
340  IF  N=3  THEN  B=B-1
350  IF  N=4  THEN  B=B+1
360  K=K+1
370  IF  K<C  THEN  GOTO  310
400  GOTO  10
```

上列程序是假设汉字由“横”、“竖”、“撇”、“捺”四个笔划组成。程序用了五个变量，其中四个变量的意义如表1所示，还有一个用于修改笔划的变量M，画笔划时要使M=1，消笔划时M=0。

将你要写的汉字，先在方格纸上写好，定出每个笔划的起始位置，然后把每个笔划的参数填入下面表1中。如“电子”两字的C、N、A、B值列于表2中。运行程序时，每输入表2中一行数据（输入一个数后，应按ENTER），即显示出一个笔划，当全部数据输入完后，显示屏上便出现“电子”两字。

4．画图样

程序1：打印出三角形图形。

表 1

C	N	A	B
行　数	1（横）	起（列）	止（列）
列　数	2（竖）	起（行）	止（行）
撇次数	3（撇）	第一撇的行	第一撇的列
捺次数	4（捺）	第一捺的行	第一捺的列

表 2

C	N	A	B	M
4	2	6	12	1
6	1	5	11	1
12	2	6	12	1
9	1	5	11	1
12	1	5	11	1
8	2	4	16	1
17	1	8	13	1
4	3	5	25	1
5	1	19	25	1
1	4	9	22	1
10	1	18	27	1
2	4	16	21	1
23	2	11	17	1

```
10 FOR I=0 TO 7
20 FOR J=7-I TO 7+I
30 PRINT AT I, J; "*"
40 NEXT J
50 NEXT I
```

程序 2：打印出平行四边形图形。

```
10  FOR  X=0  TO  7
20  PRINT  TAB(7-X); "*****"
30  NEXT  X
```

把20行改为

```
20  PRINT  TAB(7-X); "*****"; TAB(12+X); "*****"
```

试运行一次，观察结果。

程序3：画出以(31，21)为中心的十个大小不同的方块。

```
10  FOR  N=1  TO  20  STEP  2
20  GOSUB  100
30  NEXT  N
40  STOP
100  S=N
110  FOR  L=-S  TO  S
120  PLOT  31+L, 21-S
130  PLOT  31+S, 21+L
140  PLOT  31-L  21+S
150  PLOT  31-S, 21-L
160  NEXT  L
170  RETURN
```

5. 画正弦函数曲线

```
10  FOR  N=0  TO  59
20  PLOT  N, 22+20*SIN(N*π/30)
30  NEXT  N
```

这个程序运行后，将显示出由60个点子组成的正弦曲线，每个点子的角度差为6°。

实验十　游戏和音乐

一、实验目的

1. 熟悉INKEY $（读键符）函数及PAUSE（暂停）语句的作用。

2. 初步了解游戏程序和音乐程序。

二、实验内容

1. 了解INKEY $ 函数的作用

```
10  PRINT  INKEY $;
20  GOTO  10
```

运行程序后（打入RUN，按ENTER键），随便按什么键，立即把你所按键的键符显示出来。按键时间稍停片刻，则很快地显出一串键符。为了减慢程序运行速度，可加插下面语句：

```
15  PAUSE  5
```

再试验一次，观察实验结果。

2. “满天星”和“蝴蝶纷飞”的程序

```
10  K $ = INKEY $
20  I = INT(21 * RND)
30  J = INT(31 * RND)
40  PRINT  AT  I, J; K $
60  GOTO  10
```

这个程序中，I是0～21整数的随机值，J是0～31整数的随机值，它们分别代表打印位置的行数和列数。如按键的键符是星号“ * ”，不断按下此键，则显示屏上出现“满天星”。如键符是蝴蝶，则出现“蝴蝶纷飞”。

插入第50行和第55行内容，重新运行程序，并观察其结果。

```
50 PAUSE 10
55 PRINT AT I, J; "␣"
```

3．利用INKEY $ 函数画图

```
10 I=11
20 J=16
30 K $ = INKEY $
40 IF K $ = "C" THEN I=I-1
50 IF K $ = "V" THEN I=I+1
60 IF K $ = "B" THEN J=J-1
70 IF K $ = "N" THEN J=J+1
110 PRINT AT I, J; "*"
120 GOTO 30
```

运行此程序后，按一次C键使“＊”上移1格。按一次V键使“＊”下移1格。按一次B键使“＊”左移1格，按一次N键使“＊”右移1格。

如果搜索键符的速度太快，可以加插如下的语句：

```
35 PAUSE 10
```

如果I和J值容易超出范围，可以加插如下语句：

```
75 IF I<0 THEN I= 0
80 IF I>21 THEN I=21
85 IF J<0 THEN J= 0
90 IF J>31 THEN J=31
```

如果“＊”号移动得太快，可以加插如下语句，将“＊”消迹后再显示，这样就使“＊”号变成闪动的了。

```
113 PRINT AT I, J; "␣"
```

```
116 PAUSE 5
118 PRINT AT I, J; "*"
```

通过控制C、V、B和N键来控制"*"号上下左右移动，可以画出一些有规则的图形。

4．"万花筒"游戏程序

```
10 A=π/12
20 FOR E=10 TO I STEP—I
30 FOR B=0 TO 23
40 X=11+0.2*E*COS(B*A)
50 Y=15+0.5*E*SIN(B*A)
60 PRINT AT X, Y; "⋈"
70 NEXT B
80 SCROLL
90 NEXT E
100 GOTO 20
```

打入本程序并运行后，在屏幕上显示出不断变化的类似万花筒的图样。

5．音乐程序

```
10 A$="A3G1A1C>1A1G1E2G1E1D3E1G3E1D1E1C1B<1A<2C1B<1A<4D3E1A2A1G1E1G1D1E1C3D1G3E1D1E1C1B<1A<2C1B<1A<4"
20 FOR I=1 TO 5
30 TEMPO 100/I
40 MUSIC A$
50 NEXT I
60 GOTO 20
```

这个程序将歌曲由慢变快唱五遍，再反复循环不断唱下

去。如要它停止，可按下BREAK键，使程序中断执行。

实验十一 计算机与磁带机的连接——程序的存贮和程序的装入

一、实验目的

1. 了解利用盒式磁带存贮程序的方法。

2. 学会程序贮存和程序装入的操作方法。

二、操作步骤

当计算机断电以后，贮存在内存贮器中的程序和数据会全部丢失。为了使程序和数据能够长期地保存起来，需要有一个外存贮器。最简单的外存贮器就是利用家庭用的盒式录音机，把程序或数据记录在盒式磁带上，这个工作步骤称为程序的贮存。在需要使用的时候，再把贮存到盒式磁带上的程序和数据送回到计算机中，此时称为程序的装入。

计算机与磁带机连接时，磁带机有脱机和联机两种工作状态。脱机是指磁带机脱离计算机的控制，它完全由使用者去控制磁带机的起动和停止；而联机则是磁带机受计算机的控制，这时，计算机是通过专用键（或是REMOTE插孔）去控制磁带机的启停。

通常使用家用录音机作为磁带机时，均是在脱机状态下工作的。

1．贮存程序的操作步骤

（1）将计算机和录音机连接起来。用计算机配套的一对导线，把计算机后面的MIC插孔和盒式录音机的MIC插孔连接起来。此时最好不连接两机的EAR插孔，因为它会使计算机的信号失真。

（2）将磁带放入录音机中，利用快进键或快退键，把磁带绕到开始存入程序的位置，并记下录音机中计数器的读数。

（3）在计算机上键入：

SAVE“程序名”（不要按ENTER键）

在R1微型机上，使用的程序名可以是任意长度的任何名字，如“ABC”或程序的起止行号"10 TO 200"均可作为程序名，但程序名一定要用引号括起来，同时程序名不能省略。

（4）按下录音机RECORD（记录）与PLAY（放音）两键，稍等几秒钟，让磁带走速稳定下来。

（5）按下计算机的ENTER键，即进入贮存程序的状态。按下ENTER键后，开始约5秒钟内显示屏上出现跳动的白细条纹。当显示屏出现黑白粗斜线条时，标志着进入贮存程序的状态，此时程序正在被记录在磁带上。当显示屏变黑，并在右下角位置上出现OK信息时，表示程序已记录完毕。

（6）停止录音机，把磁带倒回到起始位置。

（7）检查磁带是否录上程序：可用在放音状态下使磁带再运行一次，听磁带声音的方法来检查。如磁带声音正常，没有背景噪声，则初步认为记录正常。

在贮存程序时，影响程序的贮存质量的最关键的因素是选择高质量的磁带，其次是把两机连接好。

2. 装入程序（或读入程序）的操作步骤：

（1）用计算机配套的导线，将计算机的EAR插孔和录音机的EAR插孔连接起来。

（2）利用快进键或快退键，根据计数器的指示，找到所

需程序的开始前的位置上。

调整录音机的音量至最大值的3/4左右，并把高音调大，低音调小，因为计算机的信号都是高音的。

（3）启动计算机，键入NEW命令，清内存程序。然后键入。

LOAD"　　　　"（不按ENTER键）

在装入程序时，可不打入程序名，但要键入引号和引号内的空格。

（4）按下录音机的PLAY键。要注意：什么时候按下此键是很关键的一步，必须准确地在磁带刚好绕到程序前的空白地方时按下。否则就不是从程序开始处装入，而是在程序中间装入，结果会使计算机处于自锁状态，程序不能输入到计算机。

（5）按ENTER键，进入装入程序的状态。开始的几秒钟，显示屏出现不规则的黑斜线，然后出现粗大的棒状线条上下跳动的图象，此时表明计算机正在从磁带上读出数据。如果得不到这种图象，可能是由于录音机的音量太大或太小的缘故。这时要调整录音机的音量电位器，使显示屏出现粗棒线条时为止。

当显示屏左下角出现OK字样时，装入程序工作便告结束。

（6）键入LIST命令，让程序显示出来，检查装入程序工作的质量。

影响程序的装入的因素很多，可归纳为如下几点：

（1）要使用高质量的录音带。

（2）磁头要十分清洁。使用前一定要认真清洁磁头，以后每四小时清洁一次，并且要定期退磁。

（3）磁带的倒带与按下PLAY键时，要准确地对准程序开始前的空白处。

（4）音量调节要适当。如音量太小会导致某些信号失真。音量过大也会使信号失真，并产生其它干扰信号。

附录一　习题参考答案

〔2-1〕　分三个步骤：打入NEW命令，清内存；输入程序；打入RUN命令，运行程序。

〔2-2〕　按ENTER键。

〔2-3〕　设是A键。如只按A键，可得到A键符。同时按SHIFT键和A键，可得到该键上方所示的键符。进入GRAPHIC状态后，按A键和同时按SHIFT和A键，又得到另两种键符。

〔2-4〕　打入LIST，按ENTER键；打入LISTn，按ENTER键。

〔2-5〕　打入想要消去语句的行号n，按ENTER键。

〔2-6〕

（1）　8.6573422E+9

（2）　−1.23E−7

〔2-7〕

（1）　漏括号，应为X**(Y−1)。

（2）　漏括号，应为X**(2*X)*Y。

（3）　64°要化为弧度，应为COS(64*π/180)。

（4）　漏*号，应为A+B*C。

〔2-8〕

（1）　打入PRINT　5+3*SQR(2.4*2.4+1)，按ENTER键。

（2）　打入PRINT　(286−64)*(2**6)/(1.2*EXP4)，按ENTER键。

〔3-1〕　50行语句的赋值号左边只能用一个变量，改

为X = Y - 1。

20、30行句末不能加逗号。改为INPUT A INPUT B。

10行语句的A，B之间不能用“；”改为READ A，B

80行的打印语句不能并作赋值语句。改为PRINT "X="；Y + 2。

60行语句漏PRINT，改为PRINT "X + Y"；X + Y。

〔3-2〕 (1)程序里缺少与READ语句匹配的DATA语句（没有数据）。

(2)在第30行语句中，计算S值所使用的R变量没有赋值，第40行语句的M，N变量也未赋值。

〔3-3〕 RUN

```
1    2
2    1
```

第40～60行语句中的三个赋值语句的作用是：将A和B数值相互交换。

〔3-5〕

```
10  I = 2
20  R = 25
30  U = I * R
40  ?  "U = "; U; "V"
```

〔3-6〕

```
10  INPUT  L  (单位用亨利)
20  INPUT  C  (单位用法拉)
30  F = 1/(2 * π * SQR(L * C))
40  ?   F
50  GOTO  10
```

用键盘输入L，C值：

L = 0.012　　C = 2.2E - 6

L = 0.002　　C = 1E - 6

L = 0.0004　　C = 0.002E - 6

〔3-7〕

```
10  PRINT  "F=";
20  INPUT  F
30  ?   F,
40  C=(F-32)/1.8
50  ?   "C="; C
60  GOTO  10
RUN
F=32        C=0
F=0.76      C=80
F=212       C=100
```

〔3-8〕

```
10  ?   "A␣␣␣"; "B␣␣␣";"C␣␣␣"; "D␣␣␣"; "V"
15  ?
20  INPUT  A
30  INPUT  B
40  INPUT  C
50  INPUT  D
60  V=(A+B+C+D)/4
70  ?   A; "␣␣"; B; "␣␣";C; "␣␣"; D; "␣␣"; V
```

〔3-9〕　确定算式：设鸡为X只，兔为Y只。则

$$\begin{cases} X+Y=H \\ 2X+4Y=F \end{cases}$$

解得　$X=(4H-F)/2$

$Y=(F-2H)/2$

写出如下程序：

```
10  INPUT  H
20  ?  "H="; H,
30  INPUT  F
40  ?  "F="; F
50  ?
60  X=(4*H-F)/2
70  Y=(F-2*H)/2
80  ?  "COCK="; X, "RABBIT="; Y
90  GOTO  10
```

〔4-1〕

```
1010  INPUT  A
1020  INPUT  N
1030  X= 1
1040  I=1
1050  X=X*A
1060  I=I+1
1070  IF  I<N  THEN  GOTO  1050
1080  PRINT  X
1090  STOP
```

〔4-2〕

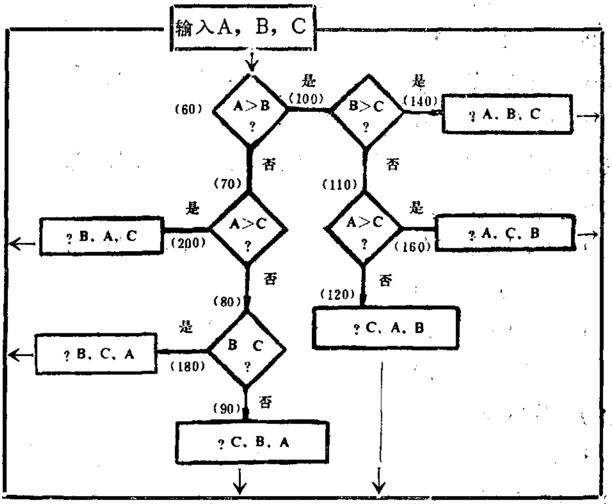
（1）设输入的三个数分别是A，B，C。则它们之间的关系共有六种情况，即

A＞B＞C，　A＞C＞B，

B＞C＞A，　B＞A＞C，

C＞A＞B，　C＞B＞A，

（2）根据上述分析，可画框图如下：



（3）根据框图编得程序如下：

```
10  INPUT  A
20  INPUT  B
30  INPUT  C
40  PRINT  "A="; A; "␣␣"; "B="; B; "␣␣"; "C="; C
50  IF  A=-1  THEN  GOTO  220
60  IF  A>B  THEN  GOTO  100
70  IF  A>C  THEN  GOTO  200
80  IF  B>C  THEN  GOTO  180
```

```
90  PRINT  C; "␣␣"; B; "␣␣"; A
95  GOTO  10
100  IF  B>C  THEN  GOTO  140
110  IF  A>C  THEN  GOTO  160
120  PRINT  C; "␣␣"; A; "␣␣"; B
130  GOTO  10
140  PRINT  A; "␣␣"; B; "␣␣"; C
150  GOTO  10
160  PRINT  A; "␣␣"; C; "␣␣"; B
170  GOTO  10
180  PRINT  B; "␣␣"; C; "␣␣"; A
190  GOTO  10
200  PRINT  B; "␣␣"; A; "␣␣"; C
210  GOTO  10
220  STOP
```

〔4-3〕

```
10  INPUT  A
20  INPUT  B
30  IF  B=9999  THEN  GOTO  60
40  IF  A<B  THEN  A=B
50  GOTO  20
60  PRINT  "MAX="; A
```

〔4-4〕

```
10  INPUT  X
20  IF  X=-1  THEN  GOTO  70
30  IF  X>=0  AND  X<10  THEN  PRINT  "Y=10"
```

```
40 IF X>=10 AND X<20 THEN PRINT "Y=20"
50 IF X>=20 AND X<30 THEN PRINT "Y=30"
60 GOTO 10
70 STOP
```

〔4-5〕 电容器放电公式为：

$$U=U_0\ e^{\frac{-t}{RC}}$$

根据题意，$t=1$时 $\frac{U}{U_0}\leqslant 0.05$

即 $e^{-\frac{1}{RC}}\leqslant 0.05$作为假定条件。

```
10 INPUT R
20 INPUT C
30 IF R=-1 THEN GOTO 80
35 T=1/(R*C)
40 K=EXP(-T)
50 IF K<=0.05 THEN PRINT "R="; R, "C="; C
70 GOTO 10
80 STOP
```

〔4-6〕

```
10 T=0
20 N=0
30 N=N+1
40 T=T+N*N
```

```
50 IF T<10000 THEN GOTO 30
60 PRINT "N="; N; "T="; T
```

〔6-1〕

(1)RUN↙

```
1 2 3 4
```

(2)RUN↙

```
0          -3
-6         -9
```

(3)RUN↙

```
1          8
Y>20
```

〔6-2〕

(1)11;

(2)6.5;

(3)-7。

〔6-3〕

(1)错；循环出现交叉。

(2)对；可用GOTO语句转出循环。

〔6-4〕

(1)RUN↙

```
  3↙      2↙
X=3         Y=2
Z=9         T=0.98768834
ST  IN  45
```

(2)RUN↙

```
  3↙   2↙
X= 3        Y= 2
```

```
Z= 9
OK IN 120
```

在(1)中，使用GOSUB-RETURN语句，执行40行语句，打印出T；在(2)中，使用GOTO语句，没有执行40行与45行语句，故不能打印出T值。

〔6-5〕

```
10 X=0
20 Y=0
30 FOR I=1 TO 50
40 INPUT T
50 IF T<=X THEN GOTO 70
60 X=T
70 IF T>=Y THEN GOTO 90
80 Y=T
90 NEXT I
100 ? "MAX="; X, "MIN="; Y
110 STOP
```

〔6-6〕

```
10 X=0
20 FOR I=1 TO 64
30 X=X+3**(I-1)
40 NEXT I
50 ? X
RUN↙
1.7168419E+30
```

〔6-7〕

```
10 T=0
```

```
20 INPUT N
30 IF N<=0 THEN GOTO 130
40 FOR I=1 TO N
50 INPUT X
60 ? "X("; I; ")="; X
70 T=T+X
80 ?
90 NEXT I
100 ? "N="; N
110 ? "T="; T
120 STOP
130 ? "N<=0 AGAIN"
140 GOTO 20
```

〔6-8〕

```
10 FOR N=1 TO 10
20 N1=N*N
30 N2=N1*N
40 N3=N2*N
50 N4=N3*N
60 ? N; "␣␣␣"; N1; "␣␣␣"; N2;
"␣␣␣"; N3; "␣␣␣"; N4
70 ?
80 NEXT N
```

〔6-9〕

```
10 INPUT N
20 ? "N="; N
30 T=0
```

```
40  FOR  I=1  TO  3
50  A=I*N
60  GOSUB  200
70  T=T+C
80  NEXT  I
90  ?  "TOTAL=";  T
100  STOP
200  C=1
210  FOR  J=1  TO  A
220  C=C*J
230  NEXT  J
240  RETURN
```

〔6-10〕

如果从6－4(1)中去掉STOP语句，则执行完40行语句后，将进入100～130行的子程序，再次打印出一个Z＝9，并打印“RF　IN　130”的出错信息。

如果从6－4(2)中去掉STOP语句，则对执行结果无影响，因此程序中的STOP实际上是多余的。

〔7-1〕

(1)不是。(2)不是。(3)是；一维；Z。

(4)是；二维；A，2。(5)不是。

(6)是；一维；A-B。(7)是；一维；2A＋2B＋2C。

(8)是；二维；A-B，B-C。

〔7-2〕

(1)1，1，1；

(2)5，3。

〔7-3〕RUN↙

(1) MAT A = (2, 4, 6, 8, 10)

(2) RUN↙

```
2 3  4
3 4  5
4 5  6
```

〔7-4〕

```
10  DIM  U(5)
20  FOR  N = 1  TO  5
30  U(I) = N * N + 1
40  ?  U(I); ", ";
50  NEXT  N
60  ?  "…,  N * N + 1  …"
70  STOP
RUN↙
2, 5, 10, 17, 26, …, N * N + 1, …
```

〔7-5〕

```
10  DIM  A(10, 10)
20  FOR  I = 1  TO  10
30  FOR  J = 1  TO  10
40  IF  I = J  THEN  GOTO  70
50  A(I, J) = 0
60  GOTO  80
70  U(I, J) = 1
80  ?  "␣"; U(I, J);
90  NEXT  J
100  ?
110  NEXT  I
```

```
RUN↙
    1  0  0  0  0  0  0  0  0  0
    0  1  0  0  0  0  0  0  0  0
    0  0  1  0  0  0  0  0  0  0
    0  0  0  1  0  0  0  0  0  0
    0  0  0  0  1  0  0  0  0  0
    0  0  0  0  0  1  0  0  0  0
    0  0  0  0  0  0  1  0  0  0
    0  0  0  0  0  0  0  1  0  0
    0  0  0  0  0  0  0  0  1  0
    0  0  0  0  0  0  0  0  0  1
```

〔7-6〕

```
10  DIM  A(3,3)
20  DIM  B(3,3)
30  ?  "MAT␣A"
40  ?
50  FOR  I=1  TO  3
60  FOR  J=1  TO  3
70  A(I, J)=3*(I-1)+J
80  ?  A(I, J); "␣";
90  NEXT  J
100  ?
110  ?
120  NEXT  I
130  ?
140  ?  "MAT␣B"
150  ?
```

```
160 FOR I=3 TO 1 STEP -1
170 FOR J=3 TO 1 STEP -1
180 B(I, J)=3*(I-1)+J
190 ? B(I, J); "␣";
200 NEXT J
210 ?
220 ?
230 NEXT I
240 STOP
```

〔7-7〕

```
10 DIM A(2,4)
20 DIM B(2,4)
30 DIM C(2,4)
40 ? "MAT␣A"
50 ?
60 FOR I=1 TO 2
70 FOR J=1 TO 4
80 INPUT A(I,J)
90 ? A(I,J); "␣";
100 NEXT J
110 ?
120 ?
130 NEXT I
140 ? "MAT␣B"
150 ?
160 FOR I=1 TO 2
170 FOR J=1 TO 4
```

```
180 INPUT B(I, J)
190 ? B(I, J); "␣";
200 NEXT J
210 ?
220 ?
230 NEXT I
240 ? "MAT C=MAT A+MAT B"
250 ?
260 FOR I=1 TO 2
270 FOR J=1 TO 4
280 C(I, J)=A(I, J)+B(I, J)
290 ? C(I, J); "␣";
300 NEXT J
310 ?
320 ?
330 NEXT I
340 STOP
```

〔8-1〕

(1)不成立。

(2)不成立。

(3)成立。

(4)(a)不成立；(b)成立。

〔8-2〕

(1)THERE␣ARE␣THREE␣DUCKS ␣ON␣STREAM

(2)WHAT␣IS␣YOUR␣NAME

WHAT␣IS␣HIS␣␣NAME

(3)54

(4)4　　　5728

4　　　5723

〔8-3〕

```
10 A$="——CHINA"
20 B$="——JAPAN"
30 C$="——CANADA"
40 D$="——USA"
50 INPUT N
60 IF N<>INT(N) THEN N=-1
70 IF N<1 OR N>200␣ THEN ? "THE␣NUMBER␣OF␣YOUR␣INPUTTING␣IS␣WRONG, AGAIN"
80 IF N>=1 AND N<=50 THEN ? A$
90 IF N>=51 AND N<=100 THEN ? B$
100 IF N>=101 AND N<=150 THEN ? C$
110 IF N>=151 AND N<=200 THEN ? D$
120 GOTO 50
```

〔8-4〕

```
10 DIM B$(5)
20 DIM A$(5, 5)
30 FOR I=1 TO 5
40 INPUT A$(I)
50 ? "A$(";I;")=";A$(I);
```

```
60 NEXT I
70 INPUT B$
80 FOR I=1 TO 5
90 IF B$=A$(I) THEN ? A$(I); "'
S␣FILE␣IS␣IN␣"; I
100 NEXT I
110 GOTO 70
```

〔9-1〕

(1) $14 \leqslant X < 15$

(2) $3.815 \leqslant X < 3.825$

(3) $3.9 \leqslant X < 4.0$

(4) $-2.05 < X \leqslant -1.95$

〔9-2〕

(1) 10.65 20.58 13.24

〔9-3〕

```
10 FOR I=1 TO 100
20 X=INT(RND*100)
30 ? X; "␣␣";
40 NEXT I
50 STOP
```

〔9-4〕

```
10 FOR I=1 TO 6
20 ? TAB (5*(I-1)+3); I-1;
30 NEXT I
40 ?
50 ?
60 FOR I=1 TO 6
```

```
70 ?  I-1;
80 FOR J=1 TO 6
90 X=((I-1)*10+J-1)**(1/3)
100 Y=INT(X*100+0.5)/100
110 ?  TAB (5*(J-1)+3); Y;
120 NEXT J
130 ?
140 ?
150 NEXT I
```

〔9-5〕

(1)

```
10 FOR I=1 TO 10
20 FOR J=1 TO I
30 ?  TAB(J-1); "*";
40 NEXT J
50 ?
60 NEXT I
```

(2)

```
10 FOR I=1 TO 10
20 FOR J=10-I TO 10+I
30 ?  AT I, J; "*"; AT 20-I, J; "*";
40 NEXT J
50 ?
60 NEXT I
```

〔9-6〕

```
10 FOR I=32 TO 99
20 A=I*I
```

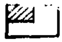
```
30  A1=INT ( A/1000 )
40  A2=INT ( (A - A1 * 1000)/100 )
50  A3=INT ( (A - A1 * 1000 - A2 * 100)/10 )
60  A4=A - A1 * 1000 - A2 * 100 - A3 * 10
70  IF  A3=A4  AND  A2=A1  THEN  ? I, A
80  NEXT  I
```

〔9–7〕

```
10  DIM  X ( 30 )
20  DIM  Y ( 30 )
30  FOR  I=1  TO  30
40  X ( I ) =INT ( RND * 100 ) +100
50  ?  X ( I ) ; "␣␣";
60  NEXT  I
70  ?
80  FOR  J=1  TO  30
90  ?  X ( J ) ;  "=";
100  FOR  I=2  TO  SQR  X(I)
110  IF  X ( J ) /I<  >INT(X ( J ) /I)  THEN
GOTO  150
120  X ( J ) =X ( J ) /I
130  ?  I;  "*";
140  GOTO  110
150  NEXT  I
160  ?  X ( J )
170  NEXT  J
180  STOP
```

附录二　R1微型机字符表

下表列出R1微型机采用的全部字符及其对应的码（用十进制和十六进制表示）。

十进制码	字　符	十六进制码
0	SPACE	00
1		01
2		02
3		03
4		04
5		05
6		06
7		07
8		08
9		09
10		0A
11	"	0B
12		0C
13	$	0D
14		0E
15		0F
16	(	10
17	)	11
18	>	12

（续上表）

十进制码	字　符	十六进制码
19	<	13
20	=	14
21	+	15
22	-	16
23	*	17
24	/	18
25	;	19
26	,	1A
27	.	1B
28	0	1C
29	1	1D
30	2	1E
31	3	1F
32	4	20
33	5	21
34	6	22
35	7	23
36	8	24
37	9	25
38	A	26
39	B	27
40	C	28
41	D	29

（续上表）

十进制码	字　符	十六进制码
42	E	2A
43	F	2B
44	G	2C
45	H	2D
46	I	2E
47	J	2F
48	K	30
49	L	31
50	M	32
51	N	33
52	O	34
53	P	35
54	Q	36
55	R	37
56	S	38
57	T	39
58	U	3A
59	V	3B
60	W	3C
61	X	3D
62	Y	3E
63	Z	3F
64	THEN	40

（续上表）

十进制码	字　符	十六进制码
65	TO	41
66	STEP	42
67	RND	43
68	INKEY $	44
69	PI	45
70	未　用	46
71	未　用	47
72	未　用	48
73	未　用	49
74	未　用	4A
75	未　用	4B
76	未　用	4C
77	未　用	4D
78	未　用	4E
79	未　用	4F
80	未　用	50
81	未　用	51
82	未　用	52
83	未　用	53
84	未　用	54
85	未　用	55
86	未　用	56
87	未　用	57

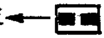
（续上表）

十进制码	字　符	十六进制码
88	未　用	58
89	未　用	59
90	未　用	5A
91	未　用	5B
92	未　用	5C
93	未　用	5D
94	未　用	5E
95	未　用	5F
96	未　用	60
97	未　用	61
98	未　用	62
99	未　用	63
100	未　用	64
101	未　用	65
102	未　用	66
103	未　用	67
104	未　用	68
105	未　用	69
106	未　用	6A
107	未　用	6B
108	未　用	6C
109	未　用	6D
110	未　用	6E

（续上表）

十进制码	字　符	十六进制码
111	未　用	6F
112	光标上↑	70
113	光标下↓	71
114	光标左←	72
115	光标右→	73
116	GRAPHICS	74
117	EDIT	75
118	ENTER	76
119	DELETE	77
120	LMODE	78
121	BREAK	79
122	LINE NO	7A
123	未　用	7B
124	未　用	7C
125	未　用	7D
126	数(number)	7E
127	光标(cuvsor)	7F
128		80
129		81
130		82
131		83
132		84
133		85

（续上表）

十进制码	字　符	十六进制码
134		86
135		87
136	反 	88
137		89
138		8A
139	反 "	8B
140	反 	8C
141	反 $	8D
142	反 	8E
143	反 	8F
144	反 (	90
145	反)	91
146	反 >	92
147	反 <	93
148	反 =	94
149	反 +	95
150	反 —	96
151	反 *	97
152	反 /	98
153	反 ;	99

（续上表）

十进制码	字 符	十六进制码
155	反.	9B
156	反 0	9C
157	反 1	9D
158	反 2	9E
159	反 3	9F
160	反 4	A0
161	反 5	A1
162	反 6	A2
163	反 7	A3
164	反 8	A4
165	反 9	A5
166	反A	A6
167	反 B	A7
168	反C	A8
169	反D	A9
170	反E	AA
171	反F	AB
172	反G	AC
173	反H	AD
174	反 I	AE
175	反 J	AF
176	反K	B0
177	反L	B1

（续上表）

十进制码	字　符	十六进制码
178	反M	B2
179	反N	B3
180	反O	B4
181	反P	B5
182	反Q	B6
183	反R	B7
184	反S	B8
185	反T	B9
186	反U	BA
187	反V	BB
188	反W	BC
189	反X	BD
190	反Y	BE
191	反Z	BF
192	CODE	C0
193	VAL	C1
194	LEN	C2
195	SIN	C3
196	COS	C4
197	TAN	C5
198	ASN	C6
199	ACS	C7
200	ATN	C8

（续上表）

十进制码	字　符	十六进制码
201	LOG	C9
202	EXP	CA
203	INT	CB
204	SQR	CC
205	SGN	CD
206	ABS	CE
207	PEEK	CF
208	USR	D0
209	STR $	D1
210	CHR $	D2
211	NOT	D3
212	AT	D4
213	TAB	D5
214	* *	D6
215	OR	D7
216	AND	D8
217	<=	D9
218	>=	DA
219	<>	DB
220	TEMPO	DC
221	MUSIC	DD
222	SOUND	DE
223	BEEP	DF

（续上表）

十进制码	字 符	十六进制码
224	NOBEEP	E0
225	LPRINT	E1
226	LLIST	E2
227	STOP	E3
228	SLOW	E4
229	FAST	E5
230	NEW	E6
231	SCROLL	E7
232	CONT	E8
233	DIM	E9
234	REM	EA
235	FOR	EB
236	GOTO	EC
237	GOSUB	ED
238	INPUT	EE
239	LOAD	EF
240	LIST	F0
241	LET	F1
242	PAUSE	F2
243	NEXT	F3
244	POKE	F4
245	PRINT	F5
246	PLOT	F6

（续上表）

十进制码	字　符	十六进制码
247	RUN	F7
248	SAVE	F8
249	RAND	F9
250	IF	FA
251	CLS	FB
252	UNPLOT	FC
253	CLEAR	FD
254	RETURN	FE
255	COPY	FF

附录三　ASCII代码

八单位码	字　符	对应五单位码	字　符
000	NULL	00	无效
001	CTRLA	32	@
002		视作 NULL 字符	
003			
004			
005			
006			
007	铃	10（输出时）	换行
010		视作 NULL 字符	
011			
012	换行	10（输出时）	换行
013		视作无效	
014			
015	回车	02	回车
016		视作无效	
017			
020			
021			
022			
023			
024			
025			
026			
027			
030	CTRLX	23	#

八单位码	字　符	对应五单位码	字　符
031		视作无效	
032			
033	ESC	10（输入时）	换行
034		视作无效	
035			
036			
037			
040	空格	04	空格
041	!	视作空格字符	
042	”	24	'
043	#	23	#
044	$	视作空格字符	
045	%		
046	&		
047	'	24	'
050	(	36	(
051	)	11	)
052	*	22	*
053	+	21	+
054	,	06	,
055	−	30	−
056	.	07	.
057	/	27	/
060	0	15	0

（续上表）

八单位码	字　符	对应五单位码	字　符	八单位码	字　符	对应五单位码	字　符
061	1	35	1	110	H	05	H
062	2	31	2	111	I	14	I
063	3	20	3	112	J	32	J
064	4	12	4	113	K	36	K
065	5	01	5	114	L	11	L
066	6	25	6	115	M	07	M
067	7	34	7	116	N	06	N
070	8	14	8	117	O	03	O
071	9	0	9	120	P	15	P
072	:	16	:	121	Q	35	Q
073	;	16	;	122	R	12	R
074	<	26	<	123	S	24	S
075	=	17	=	124	T	01	T
076	>	13	>	125	U	34	U
077	?	17	=	126	V	17	V
100	@	32	@	127	W	31	W
101	A	30	A	130	X	27	X
102	B	23	B	131	Y	25	Y
103	C	16	C	132	Z	21	Z
104	D	22	D	133	[	36	(
105	E	20	E	134	\	27	/
106	F	26	F	135	]	11	)
107	G	13	G				

（续上表）

八单位码	字　符	对应五单位码	字　符	八单位码	字　符	对应五单位码	字　符
136	↑	05	↑	157	o	03	O
137	←	26	<	160	p	15	P
140	'（重音）	24	'	161	q	35	Q
141	a	30	A	162	r	12	R
142	b	23	B	163	s	24	S
143	c	16	C	164	t	01	T
144	d	22	D	165	u	34	U
145	e	20	E	166	v	17	V
146	f	26	F	167	w	31	W
147	g	13	G	170	x	27	X
150	h	05	H	171	y	25	Y
151	i	14	I	172	z	21	Z
152	j	32	J	173	{	36	(
153	k	36	K	174	\	27	/
154	l	11	L	175	}	11	)
155	m	07	M	176	~		
156	n	06	N	177	DEL		

附录四　R1微型机函数和语句一览表

1．函数

功能字	键入参数(X)	结果
ABS	数	绝对值
ACS	数	反余弦，以弧度为单位 当X不在－1到＋1之间，出错报告为AG
ASN	数	反正弦，以弧度为单位 当X不在－1到＋1之间，出错报告为AG
ATN	数	反正切，以弧度为单位
CHR$	数	读出编码为X的字符，取近似整数。 如X不在0到255之间，出错报告为IR
CODE	字串	读出X字串第一个字符的编码（如X为空字串则为0）
COS	数	余弦，以弧度为单位
EXP	数	e^X
INKEY$	无	读键符。如果按下的键是有字符的话，则函数值为键所代表的字符（正常状态）。如键没有字符则为空字串

功能字	键入参数(X)	结　　果
INT	数	取不大于X值的最大整数
LEN	字串	字串长度（如为空字串则为0）
LON	数	自然对数（底为e） 如X≤0时，出错报告为AG
PEEK	数	读出地址为X的内存里的字节的值（近似整数）。 如X不在0到65535之间，出错报告为IR
PI	无	π(3.14159265)
RND	无	伪随机数
SGN	数	X的符号值（−1，0或+1）
SIN	数	正弦函数值，以弧度为单位
SQR	数	平方根 如X<0,出错报告为AG
STR$	数	读出数X的字符串
TAN	数	正切函数值，以弧度为单位
USR	数	调用机器码程序段，其起始地址为X（近似的整数），返回地址为BC寄存器对的内容。 如X不在0到65535范围

功能字	键入参数(X)	结　　　　果
		时，出错报告为 I R
VAL	字串	函数值为字串的数字表达式。如X的内容不能形成有效的数字时，则给出报告信息IE；或根据表达式给出其他的出错报告

2．语句

语句中表达式采用的符号的说明：

a	表示一个字母
v	表示一个变量
X、Y、Z	表示数字表达式
m、n	表示数字表达式，取整数值
e	表示一个表达式
f	表示字串值表达式
s	表示一个语句

语句中除INPUT语句外，均可作为命令使用。除了语句的标号外，都允许使用表达式。

BEEP	进入发声状态，按键有声音。使输入程序行、打入命令，或键盘输入数据(INPUT)时，均会发出不同频率的声音
CLEAR	清除所有变量，空出它们占用的空间。
CLS	(Clear Screen)清除显示屏上的画面
CONT	如报告信息是ST IN YY，打入CONT命令后，从YY下一个语句继续执行。如报告信息是BK IN YY，则CONT命令，相当于GOTO YY

COPY	外接打印机时使用。作用是传送一个文件给打印机。COPY命令无先清屏的功能
DIMA$(n_1, \cdots, n_k)$	定义K维数组最大下标值（下标值从1计起），并对所有A下标变量清零。如果内存空间不足存放这个数组时，给出OM报告信息。使用下标变量，必须要有此语句
DIMA$ $(n_1, \cdots, n_k)$	定义长度为n_k的K－1维的字串组。所有字串的始值定为“ ”。如内存空间不足以存放这个字串组时，给出OM报告信息。引用字串组时，要加DIM语句
FAST	进入快状态。在快状态期间停止显示画面。等到暂停或程序结束后，或执行INPUT时，才将打印内容全部显示出来
FOR A＝X TO Y	FOR A＝X TO Y STEP 1
FOR A＝X TO Y STEP Z	清变量A，确定循环变量始值为X，终值为Y，步长值为Z。如果定义终值引起超出内存容量时，则给出OM报告信息
GOSUB N	转到入口地址为n的子程序。如果没有对应的RETURN时，给出报告信息OM
GOTO N	无条件转向n行。如程序中没有n行，则转到它的下一行
IF X THEN S	如果X条件成立，就转到执行S语句。不能用IF X THEN行号的格式

INPUT V	暂停执行程序，等待键盘输入V值。如输入值不符合规定时，给出II报告信息。INPUT语句不能作命令用。在FAST状态时，遇到INPUT语句时，有显示输出
LET v=e	给变量v赋值e，LET可省去。如一个简单变量，没有LET或INPUT语句赋值时，则它是属于没有定义的变量。如果v是字串变量或者是子字串变量时，则e应为字串值。当它的长度超出定义的长度时，只取定义长度的字串
LIST	LIST 0，从最小的行号开始，列出程序清单。程序光标指在0行
LIST n	从n行开始列出程序清单。程序光标指在n行。如程序清单太长，超出显示屏范围时，给出OM或SF报告信息
LLIST n	类似LIST n，但用了打字机代替显示屏，如未接打印机就什么都不执行
LOAD f	在磁带上找出名字为f的程序，并把它及其变量输入到计算机。如f=" "时，则输入第一个程序。当磁带读不到程序时，应按BREAK键，停止进行。如果只读入了部分程序，则应打入NEW命令清除
LPRINT	类似PRINT。但用了打印机代替显示器。一般语句末不带分号或逗号，使每次打印一行，如一行打不完，则转到下

一行继续打印

MUSIC f	音乐语句。字串f代表一系列音符。每个音符包含2到4个字符，其中第一、二个字符分别表示音符（半音由反体字符表示）和高低八音。第三、四个字符表示音符持续多少个时间单位。时间单位值由TEMPO语句确定。按RESET键后，时间单位起始值定为约0.1秒(相当于TEMPO 25） 休止符用“·”表示。允许用空格符，但对它不作处理。如字符串格式不对时，给出MF报告信息
NEW	重新使用BASIC系统。清除内存中程序和变量
NEXT A	将循环变量值加上步长值后，转到FOR语句。当步长值＞0，A＞终值（步长值＜0时，则用A＜终值）时，转到NEXT 下一个语句。NEXT语句必须与FOR语句成对使用，所用变量也应一致。如NEXT中变量（已被赋值）不是FOR中变量时，则给出NF报告信息；假如它所用变量未被赋值，则给出UV报告信息
NO BEEP	消除发声功能，使计算机进入“寂静”状态。此时按下任何键，就不再有声音了
PAUSE n	使程序暂停执行 $n\times\frac{1}{50}$ 秒的时间。但保

		留原来画面。$0 \leqslant n \leqslant 65535$，否则出错报告IR。$n \geqslant 32768$时，暂停时间直至有键按下时为止
PLOT	m,n	显示象元（pexel）在（m,n）的位置上。原点（0，0）在左下角。$0 \leqslant m \leqslant 63$，$0 \leqslant n \leqslant 43$，否则，出错报告IR。
POKE	m,n	把数值n写到地址为m的RAM存贮单元中去。$0 \leqslant m \leqslant 65535$，$0 \leqslant n \leqslant 255$，否则出错报告IR
PRINT	…	"…"为一组打印项。两项之间用逗号或分号隔开。本语句是用在电视机作为显示器时的打印语句。它有两种输出格式：分段格式和紧凑格式。分段格式采用的是逗点符号，将一个显示行分两段，每段占16格。紧凑格式采用的符号是分号，每个打印项之间不留空格。句末无分号或逗号时，则执行下一打印语句时，将自动换行。 打印项内容有如下几种： (1)空白（即什么也不显示）； (2)数字表达式：显示的数小于10^{-5}和大于10^{13}时，用指数记数法表示。如表达式含有未赋值变量时，则出现UV报告信息； (3)字串：要用引号括起来。打印时，将字串原样照印； (4)AT m，n。打印位置由m行n

	列定位。m值由上至下计算。0≤m≤21，否则出错报告SF。而n值则由左到右计算。0≤n≤31，否则出错报告IR；
	(5)TAB n。打印位置在n列位置上开始打印。如n值超过31，自动转至下一行继续打印。0≤n≤255，否则出错报告IR
RND	随机函数。用来产生0～1之间（不包含0和1）的随机数
REM…	注释语句。是非执行语句，计算机遇到此语句时，不产生任何操作。"…"为注释内容，一般是字符串，但不能是ENTER
RETURN	返回语句。如出现没有对应的GOSUB语句时，给出报告信息RG
RUN	RUN 0。从最小行号开始执行程序。清变量
RUN n	清变量后，从n行开始执行程序
SAVE f	把程序名为f的程序及其变量记录（存贮）在磁带上。不允许使用空字串作为程序名。否则出错报告为NA SAVE f 作为语句使用时，不能用在子程序中
SCROLL	把显示屏整个画面往上滚动一行，丢掉最顶上一行，空出最底下一行，作为打印位置
SLOW	使计算机处于既要计算和又要处理显示

	状态。执行程序时能不断进行显示
SOUND m,n	当计算机处于FAST状态时，SOUND语句产生频率为32500/m赫兹，而持续时间为n/65ms的声音。如处在SLOW状态时，持续时间约长四倍，且声音被调制为50赫兹。语句中：$0\leqslant m\leqslant 255$，$0\leqslant n\leqslant 65535$，否则出错报告IR。m=0时，按m=256处理；n=0时，按n=65536处理
STOP	暂停执行程序。并给出报告信息ST。打入CONT命令后，继续程序的执行
TEMPO n	确定MUSIC语句中的时间单位值为n×0.94ms。按RESET键后，自动执行"TEMPO 25"语句。$0\leqslant n\leqslant 255$，否则出错误报告信息IR。
UNPLOT m,n	消去由PLOT语句所显示的象元。其它与PLOT语句同。

附录五 报告信息表

报告的含义是计算机向使用者报告机器运行的情况。当一个命令或一个程序执行完毕后，或程序未执行完毕而被中断时，都会显示出一个报告信息，说明程序执行到那一行出现的情况。如果是命令执行完毕，则只给出报告，而不显示出行号，且一直保留到有按键时为止。

报告	意义	出现条件
OK	执行成功，或跳到比现存行号大的一行	任何情况
NF	有NEXT没有FOR语句，或NEXT语句中变量，不是FOR语句中的变量。而它的变量已被赋值	NEXT
UV	未定义的变量（Un defined variable）出现在简单变量未被赋值；下标变量没有DIM语句定义其最大下标值；NEXT语句中的变量，不是FOR语句中的变量，而用了其他未被赋值的普通变量	任何情况
BS	下标不对（Bad subscript）。如下标已超出范围（负值或大于65535），则给出报告IR	下标变量
OM	超出了内存（Out of memoay）容量	LET. INPUT，DIM，PRINT，LIST，PLOT，UNPLOT，FOR GOSUB，有时在计算时

（续上表）

报告	意　　　　　　　　义	出　现　条　件
SF	显示屏满(Screen Full)。打入CONT命令，先清屏再继续执行程序	PRINT，LIST
OV	溢出（Overflow） 计算中出现数字大于10^{38}时溢出	任何算术运算
RG	有RETURN没有对应的 GOSUB	RETURN
II	非法输入（Illegal INPUT），即把 INPUT 当作命令使用	INPUT
ST	执行了STOP语句。打入CONT继续执行下面语句	STOP
AG	对某些函数自变量不对（Invalid argnment To certain functions)	SQR，LOG，ASN，ABS，**
IR	整数超出范围(Integer out of range)。当要求整数时，浮点数会被取近似整数，如此数超出规定范围时，出现报告信息IR。X是下标变量时（参阅“BS”条内容）	RUN，RAND，POKE，DIM，GOTO，GOSUB，LIST，LLIST，PAUSE，PLOT，UNPLOT，CHR$，PEEK，USR，SOUND，TEMPO 下标变量
IE	表示方法不对（Invalid expression）。VAL后面跟随的参量(字串)的内容不能形成有效的数字	VAL

（续上表）

报告	意　　义	出 现 条 件
BK	程序被“BREAK”键所中断。在下列情况下，“SPACE”键有“BREAK”键的作用。 （a）程序执行到一个语句的结尾； （b）当计算机正在磁带上找 程序； （c）当计算机使用打印机	在任何语句的结尾，或在LOAD，SAUE，LPRINT，LLIST，COPY 或INPNT
NA	程序名定为空字串	SAVE
MF	音乐字串格式不对	MUSIC

附录六　音符字符表

字　符	音　符	频　率（赫）
C <	$\underset{\cdot}{1}$	131.0
$\boxed{C}$ <	$^{\#}\underset{\cdot}{1}$	138.3
D <	$\underset{\cdot}{2}$	147.1
$\boxed{D}$ <	$^{\#}\underset{\cdot}{2}$	155.5
E <	$\underset{\cdot}{3}$	165.0
$\boxed{E}$ <	$^{\#}\underset{\cdot}{3}$	174.7
F <	$\underset{\cdot}{4}$	174.7
$\boxed{F}$ <	$^{\#}\underset{\cdot}{4}$	184.7
G <	$\underset{\cdot}{5}$	195.8
$\boxed{G}$ <	$^{\#}\underset{\cdot}{5}$	207.0
A <	$\underset{\cdot}{6}$	219.6
$\boxed{A}$ <	$^{\#}\underset{\cdot}{6}$	233.8
B <	$\underset{\cdot}{7}$	246.2
$\boxed{B}$ <	$^{\#}\underset{\cdot}{7}$	262.1
C	1	262.1
$\boxed{C}$	$^{\#}1$	277.8
D	2	295.5

字 符	音 符	频 率(赫)
|D|	#2	312.5
E	3	331.6
|E|	#3	349.5
F	4	349.5
|F|	#4	369.8
G	5	391.6
|G|	#5	416.7
A	6	439.2
|A|	#6	471.0
B	7	492.4
|B|	#7	524.2
C	$\dot{1}$	524.2
|C|>	#$\dot{1}$	560.8
D>	$\dot{2}$	500.9
|D|>	#$\dot{2}$	625.0
E >	$\dot{3}$	663.3
|E|>	#$\dot{3}$	706.5
F >	$\dot{4}$	706.5
|F|>	#$\dot{4}$	738.6

字 符	音 符	频 率(赫)
G >	$\dot{5}$	792.7
$\boxed{G}$ >	$\#\dot{5}$	833.3
A >	$\dot{6}$	878.4
$\boxed{A}$ >	$\#\dot{6}$	955.9
B >	$\dot{7}$	984.8
$\boxed{B}$ >	$\#\dot{7}$	1048.4

果粉藏書

曹磊

封面设计：曹 磊

微型电子计算机教程

微型计算机基础 定价：1.60元

TRS—80微型计算机 BASICⅡ语言与磁盘操作系统 定价：1.60元

APPLEⅡ 微型计算机 BASIC语言与磁盘操作系统 定价：1.60元

APPLEⅡ 微型计算机 系统结构 定价：1.60元

APPLEⅡ 微型计算机 实习操作指导 定价：1.60元

R1微型计算机读本 定价：1.50元

统一书号 15182·115
定 价 1.50 元